Giampiero Bianchi

UN PERSONAL COMPUTER

MS-DOS

IMPARIAMO A PROGRAMMARE CON

M24

Zanichelli

Giampiero Bianchi

UN PERSONAL COMPUTER

IMPARIAMO A PROGRAMMARE CON M24

Zanichelli

INDICE

CAPITOLO 6 UN PO' DI MATEMATICA SU M24

CAPITOLO 7 ANCORA SULLA GRAFICA

CAPITOLO 8 IL SUONO E IL COLORE

CAPITOLO 9 LA GESTIONE DEGLI ARCHIVI

CAPITOLO 10 I SEGRETI DI BOTTEGA

APPENDICE A - UN LABORATORIO PER ESPERIMENTI MATEMATICI

PREFAZIONE

La soddisfazione che nasce dopo aver letto un libro di introduzione al Basic dotato di ricchi e ben studiati esercizi di rinforzo, distoglie l'attenzione dal fatto piu' sottile: riusciro' a calare la mia conoscenza del linguaggio su un problema nuovo -il mio problema- e a trovare la sua soluzione su una determinata macchina ?

A volte si assimila una tale quantita' di nozioni non finalizzate che capita di dimenticare lo scopo dello studio. E' come se le nozioni fossero i pezzi di una bella scatola di montaggio per costruire degli oggetti misteriosi.

Sento spesso dire da amici e conoscenti che hanno appena comperato un computer: "Non credevo fosse tanto ostico usare un calcolatore finche' non ho provato a metterci le mani sopra per risolvere anche un piccolo problema, come calcolare un'espressione algebrica o scrivere una lettera".
E ancora: "Ma se non riesco a fare con un computer quello che so gia' eseguire su strumenti piu' semplici, come posso sperare di risolvere problemi piu' difficili e complessi? "

Quello che consiglio a questi miei amici e' di non perdersi inizialmente dietro a dei problemi complessi, ma di accostarsi al nuovo mezzo per imparare a conoscerlo nelle sue prestazioni piu' semplici e per costruirsi a poco a poco un insieme di strumenti che serviranno poi per risolvere situazioni piu' complesse. Per imparare a correre bisogna prima imparare a camminare!

La metodologia adottata in questo libro e' proprio quella di cominciare subito a eseguire su M24 quello che normalmente viene fatto con strumenti meno sofisticati, come la calcolatrice tascabile e la macchina da scrivere.

Una volta insegnate queste abilita' su M24, nei capitoli e paragrafi successivi, si affronteranno problemi nuovi ben precisi, come:

-scrivere una riga sul video
-disegnare una figura sul video
-risolvere un'espressione algebrica o tracciare una curva
-spostare e animare una figura sul video
-stampare una o piu' pagine
-fondere colore, suono in ambiente grafico
-risolvere sistemi di equazioni lineari
-condurre un'analisi statistica su un campione
-etc.

Ogni capitolo contiene quindi le regole essenziali per realizzare un particolare obiettivo, partendo da un problema semplice e ben formalizzato, e si conclude con una sintesi dei parametri che possono essere variati per ottenere altri risultati nell'ambito dello stesso problema.

In particolare, il metodo qui adottato per l'apprendimento del linguaggio Basic, soddisfa nello stesso tempo a criteri di concretezza operativa e di costruttivita' formale, in modo da creare uno schema della soluzione ripetuta di problemi concreti, la cui analisi innesca un processo di costruzione generale suscettibile di essere ulteriormente generalizzato e potenziato.

In questo modo, insegnando per problemi, l'articolazione fondamentale del libro procede all'edificazione di un sistema di soluzioni di classi di problemi tra loro logicamente interconnessi ma che possono anche sussistere da soli. Ogni classe di problemi e' costituita da un programma di istruzioni, che viene presentato in forma parametrica rispetto all'istruzione fondamentale che si vuole imparare. Cosi', ogni volta che si fa girare il programma, modificando rapidamente quei parametri, si riesce subito ad esplorare il campo di validita' dell'istruzione e gli effetti delle variazioni apportate.
Si fa quindi appello costante all'inventiva e al libero sviluppo dei contenuti, secondo la tecnica del "prova e sbaglia" di galileiana memoria. Tale metodo prepara naturalmente all'indagine di programmi piu' complessi.
Infatti, per sapere come gira un programma o per capire i motivi per cui una data istruzione e' stata formulata in un certo modo, il metodo piu' efficace e', a mio avviso, quello di cambiare qualcosa dentro l'istruzione e poi osservare che cosa succede. Cosi', la costruzione astratta di enti logici inventati mediante la scrittura di un programma e le alterazioni create mediante la variabilita' di uno o piu' parametri alla volta, possono produrre effetti -a volte curiosi- la cui analisi stimola a meglio fissare l'elaborazione formale e meglio comprendere la procedura di calcolo che porta al risultato per il quale il programma e' stato scritto.

Ciascuno di questi aspetti trae inoltre maggior vantaggio ed efficacia dall'impiego intensivo della grafica di cui M24 e' ben dotato. Lo strumento utilizzato fornisce quindi una visione integrata del problema da risolvere, che puo' presentarsi come una sorta di deduzioni geometriche legate all'applicazione specifica.
Non e' soltanto un tentativo di privilegiare coloro che pensano in termini di immagini visuali geometriche, ma serve soprattutto per stimolare quelli che abitualmente ragionano per simboli.
Questa fusione di informatica e di geometria, per rendere insieme intuitivamente visibili e logicamente connesse le cause agli effetti, rende la materia piu' suggestiva e stimola la creativita', o il "far da se'". L'itinerario didattico, scelto in conformita' ai suddetti criteri informatori, conduce a costruire nuovi blocchi di istruzioni proposte dallo stesso discente che, imparato il metodo, lo puo' mettere subito in pratica.
Definiamolo pure un approccio "problem solving", o insegnamento per problemi: e' comunque un tipo di apprendimento motivato.

Il libro, in definitiva, e' rivolto:

- a chi (anche esperto di informatica) vuole apprendere rapidamente l'uso di M24 attraverso la scoperta e l'esercizio di piccoli ma significativi programmi realizzati con i "mattoni" principali del Basic,

- ai neofiti, che desiderino avvicinarsi all'informatica,

- a chi non ha mai toccato un calcolatore e vuol sapere solo quello che serve per ottenere un certo risultato.

Insomma questo e' un libro che ha la pretesa di insegnare ad usare M24 cercando di evitare la tipica crisi di rigetto e l'instaurarsi di

un ingiustificato rapporto di odio per la macchina. Inoltre, il fatto di essere un libro orientato ad uno specifico computer (M24), costituisce un altro vantaggio. Infatti, dovra' essere necessariamente preciso nei particolari operativi e privo di espressioni come: "dipende dal tipo di computer...".

In ogni caso, qualunque sia il motivo per cui Voi professionisti, docenti, giovani manager, appassionati, vi interessate al computer, lo scopo di questo libro sara' raggiunto se, anche prima di acquistare M24, saprete chiarirvi le idee su come potreste usarlo. A questo punto, sarete in grado anche di ordinarlo nella giusta e ragionata combinazione dei suoi componenti.

Con l'aiuto di questa guida, inoltre, saprete gia' se usarlo con programmi standard, fatti da altri e da comprare a parte, oppure se siete interessati a sviluppare i vostri programmi, scrivendoli personalmente.

In appendice a questo testo troverete anche un mini laboratorio per esperimenti matematici che nasce dall'insieme dei mattoni Basic appresi nei vari capitoli.

Nella seconda appendice si tracciano le linee essenziali del sistema MS-DOS per quel tanto che basta a sviluppare ambienti operativi di normale complessita'.

In conclusione, i consigli e le metodologie, in questo libro, non sono travolgenti. Non forniscono la bacchetta magica per imparare senza alcuna fatica. Chi, aldila' delle parole, arrivi a capire tutto il libro e magari riesca a leggere i programmi senza farli girare, avra' realizzato uno degli scopi al quale il libro tende. Se poi sara' anche in grado di costruire programmi simili a quelli proposti, allora avra' anche raggiunto lo scopo di toccare con mano le difficolta' di programmare un calcolatore. Soprattutto, sapra' formalizzare correttamente il problema di cui e' profondo conoscitore e potra' farlo programmare da altre persone, trasmettendo loro in modo non ambiguo l'essenziale. Il programmatore, a cui non avra' chiesto l'uso della bacchetta magica, non lo ringraziera' mai abbastanza...

E ora tocca a Voi: accendete M24 e se siete dei principianti seguite progressivamente l'itinerario proposto.

Giampiero M. Bianchi

A mia moglie.

CAPITOLO
UNO

INTRODUZIONE A M24

1.1 COME E' FATTO M24

Come ogni elaboratore elettronico, anche M24 e' fatto di hardware e software, cioe' di materiale (elettronico e meccanico) e di programmi che ne permettono un opportuno funzionamento.

Per gli scopi pratici cui questo libro e' rivolto, esamineremo in modo dettagliato solo le parti strettamente necessarie alla comprensione della struttura di M24, tralasciando per il momento quelle che, pur facendo parte dell'impianto -o potrebbero farne parte- non presiedono a funzioni esterne di immediato interesse per l'operatore.

Cominceremo a definire l'impianto, o, come si e' soliti chiamare, il **sistema**. Le sue parti principali sono:

- unita' centrale
- tastiera
- unita' video
- unita' a floppy disk
- stampante

L'unita' centrale coordina il funzionamento di tutto il sistema. E' a sua volta costituita di un organo di calcolo e di un organo di memoria.

L'M24, come ogni elaboratore di certe dimensioni, ha due tipi di memoria. La memoria principale, rapidissima e contenuta nell'unita' centrale a stretto contatto con il cuore della macchina, ed un'altra ausiliaria meno rapida, ma molto capace. La prima, la cui dimensione non supera solitamente il mezzo milione di caratteri, e' chiamata memoria R.A.M. (Random Access Memory), e' normalmente utilizzata per immagazzinare il programma di calcolo da eseguire, e anche certi risultati intermedi a reimpiego immediato che non devono essere necessariamente conservati a lungo.

Le unita' a disco -flessibile e removibile (floppy), o rigido (hard)- appartengono, invece, all'altro tipo di memoria. A seconda del tipo, queste unita' meccaniche possono contenere da 160000 ad alcuni milioni di caratteri, in relazione anche alle tecniche e alla densita' di registrazione.

A questo punto avrete gia' intuito che le informazioni possono anche essere prelevate dalla memoria, oltre che immesse. Le occa- sioni e i motivi per farlo variano non soltanto per la necessita' di riprendere certi risultati intermedi che erano stati accantonati perche' di non immediato utilizzo, ma anche per la necessita' di far eseguire parti complementari di calcolo in aggiunta a quella principale. Per inciso, ricordiamo che la memoria principale conserva l'informazione che vi abbiamo registrato solo se la macchina e' accesa; al suo spegnimento, il contenuto va perso.

Il risultato dell'elaborazione deve poi essere portato all'esterno della memoria ed evidenziato su particolari periferiche: l'unita' video e/o la stampante.

Il video, simile allo schermo di un televisore, e', al contrario, mezzo di comunicazione nelle due direzioni tra operatore e sistema. Ogni dato introdotto puo' essere seguito a vista ed eventualmente corretto. Il video consente di visualizzare un massimo di 2000 caratteri, (cifre numeriche, lettere dell'alfabeto, segni d'interpunzione...), distribuiti in 25 righe da 80 caratteri. Questa capacita' va riferita a caratteri fermi e contemporaneamente contenuti sullo schermo, ma con opportune manovre e' possibile far scorrere le righe verso l'alto del video.

Tale movimento, detto **scrolling**, consente ovviamente di estrarre dalla memoria una sterminata sequenza di righe da 80 caratteri. In tal modo abbiamo evitato di fare un video gigante con capacita' pari a quella della memoria principale, che possiamo comodamente leggere attraverso una piccola finestra di 2000 caratteri.

1.2 I COMPONENTI FONDAMENTALI

Vediamo ora un po' piu' nel dettaglio i componenti essenziali della macchina introdotti sommariamente nella presentazione generale di M24. Facendo riferimento all'assetto meccanico in cui sono assemblati e allo schema semplificato di figura 1.1, notiamo che tutti i componenti sono tra loro collegati attraverso un canale di comunicazione interno -detto anche **bus**- che in particolare consente all'unita' centrale di coordinare il funzionamento di tutti i componenti stessi del sistema.

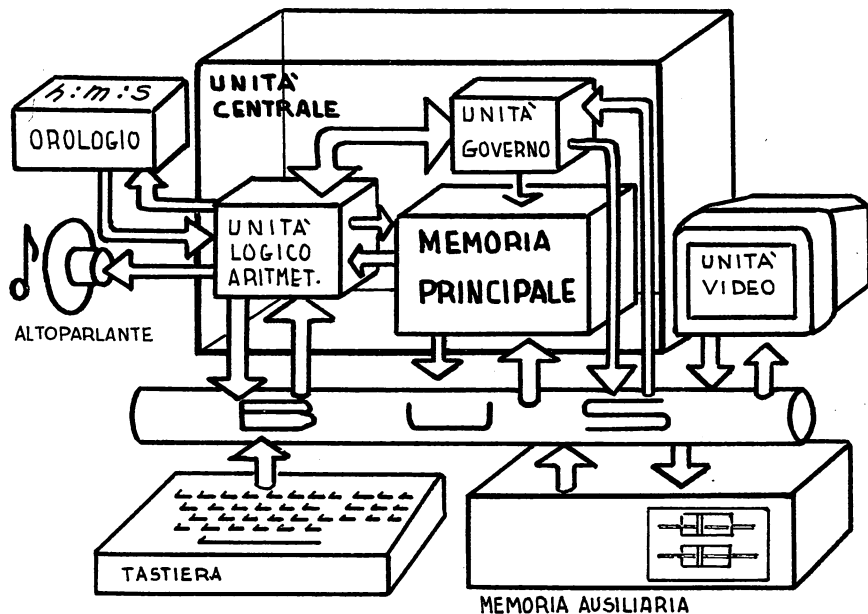



Figura 1.1

Esistono anche altre unita' periferiche che qui non consideriamo per semplicita'. Ricordiamone solo alcune: il collegamento, via accoppiatore acustico, a linee telefoniche per unire M24 ad altri M24, o a strumenti di misura, e quello per comunicare con elaboratori di grandi dimensioni, fungendo da terminale intelligente.

1.2.1 L'UNITA' CENTRALE

L'unita' centrale e' chiamata anche **CPU**, dall'inglese Central Processing Unit. Come prima detto, incorpora gli organi piu' pregiati della macchina tra i quali due meritano un particolare rilievo: l'unita' logico-aritmetica che esegue le corrispondenti operazioni, oltre che controllare il funzionamento di tutto il sistema, e l'unita' di memoria principale ad accesso veloce nella quale possono essere registrati dati e istruzioni.

In termini di capacita' di immagazzinamento, l'unita' base di M24 possiede una memoria di 128 kB (chilobyte), dove 1 kB equivale a 1024 byte, e un byte a 8 cifre binarie (bit).

Non e' facile per ora esprimere praticamente l'equivalente operativo di una tale capacita' di memoria. Teniamo tuttavia presente che un byte, costituito di 8 cifre binarie, puo' contenere 256 possibili combinazioni di 0 e 1, a ciascuna delle quali si fa corrispondere un simbolo (numeri, lettere alfabetiche, segni speciali e anche caratteri non stampabili, ad uso interno della macchina). Cosi' in 1024 byte possono essere registrati 1024 caratteri stampabili, pari a circa mezza cartella dattiloscritta.

L'unita' base di memoria e' espandibile con moduli addizionali di 128 kB, fino a raggiungere una capacita' complessiva di 640 kB.

Sul piano logico, si puo' pensare che la memoria sia come un enorme deposito di celle contigue, ciascuna selezionabile con un indirizzo assegnato. Come abbiamo gia segnalato, la proprieta' di conservare l'informazione della memoria centrale si limita al solo periodo in cui la macchina e' accesa. Al suo spegnimento, il contenuto informativo si volatilizza e di cio' bisogna tener conto per evitare la perdita di preziosi risultati o di masse di dati introdotti e non ancora trasferiti su supporti di registrazione permanenti.

A contenuto non labile, e quindi capace di mantenere le informazioni anche dopo lo spegnimento della macchina, esiste un'unita' di memoria veloce, detta **ROM** (dall'inglese Read Only Memory): tale memoria contiene alcuni programmi di servizio vitali destinati alle prime necessita' di M24, tra le quali la diagnosi preventiva di tutti i suoi organi.

All'unita' centrale sono anche collegate due piccole ma utili periferiche. Una e' l'altoparlante per la generazione dei suoni, l'altra e' un orologio che scandisce il tempo in ore, minuti e secondi. Entrambi questi dispositivi sono sotto il controllo dell'unita' centrale.

Come abbiamo prima osservato la CPU, insieme con i vari componenti e' collegata al **bus** per consentire il trasferimento di tutte le informazioni da e verso le periferiche in tutte le possibili combinazioni.

Ad esempio, cio' che compare sul video puo' essere il contenuto di una certa parte di memoria principale. Nel bus passa anche il risultato di un certo calcolo svolto dall'unita' logico-aritmetica e che viene registrato in un'altra parte della memoria centrale. Si intuisce che deve esserci a livello superiore un'ente che coordina tutte le attivita' dei vari componenti, disciplinando il traffico delle diverse informazioni.

In sintesi, ad altissima velocita', ma uno alla volta, ogni compito viene svolto e gestito in sequenza dall'unita' periferica interessata, sotto il coordinamento dell'unita' centrale.

Avrete gia' compreso che una delle risorse piu' pregiate della macchina e' la memoria centrale: tanto piu' e' grande, tanto piu' complessi possono essere i programmi che M24 puo' elaborare.

1.2.2 LA TASTIERA

Esistono due tipi di tastiere collegabili a M24: il tipo 1, schematizzato in figura 1.2 e' quello standard internazionale, mentre il secondo (figura 1.3) rappresenta la versione approntata per gli utenti M20, che cosi' salvano buona parte della manualita' acquisita su questa macchina. Sostanzialmene, pero', entrambi i tipi sono simili alla tastiera di una macchina per scrivere e possono essere ordinati alla fabbrica con una disposizione di tasti selezionata a seconda della nazione in cui vengono impiegati. Ad esempio, nella tastiera francese la lettera Q e' la prima in terza fila, mentre in tutte le altre 15 tastiere nazionali e' in seconda fila; le tastiere norvegese e greca hanno i loro caratteri speciali e cosi' via.

Per semplicita' di esposizione, nel seguito, ci riferiremo sempre al tipo 1.

Come primo impatto, una delle cose piu' evidenti che notiamo, rispetto a una macchina per scrivere, e' la presenza sulla destra di un gruppo di tasti, tra i quali le dieci cifre decimali da 0 a 9.

Tali cifre sono anche ripetute nella prima fila sopra la tastiera alfabetica. Questa disposizione consente di operare piu' rapidamente, quando il lavoro di introduzione di dati e' costituito di una grande quantita' di numeri.

Altri tasti di immediata rilevanza sono quelli cosiddetti speciali come quelli a sinistra disposti in doppia fila e numerati da F1 a F10 e quelli da usare in accoppiata con altri. Tra questi ultimi CTRL (CONTROL), ALT (ALTERNATE) e ↑ (SHIFT) (ripetuto anche sulla destra in basso) se premuti insieme con un qualsiasi altro tasto che non sia speciale, comunicano alla CPU un carattere diverso. Ad esempio il tasto SHIFT, simile all'analogo delle maiuscole sulle macchine per scrivere, se premuto con una qualsiasi lettera alfabetica, provoca il passaggio dal minuscolo al maiuscolo.

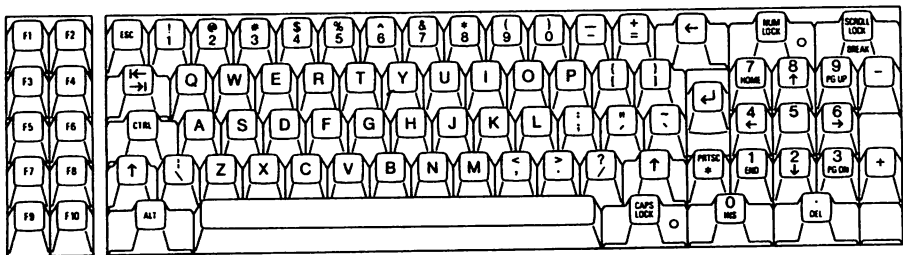
Il complesso delle combinazioni di caratteri generabili con le accoppiate tra tasti generici e i tre tasti speciali ci porta subito a considerare le enormi possibilita' a disposizione. Trascurando per il momento i 10 tasti funzionali, i 13 tasti del gruppo numerico, gli 8 tasti operativi e il tasto lungo (per lo spazio), rimangono i 47 tasti della parte alfanumerica di cui 26 con lettere dell'alfabeto

USA ASCII - TASTIERA 1

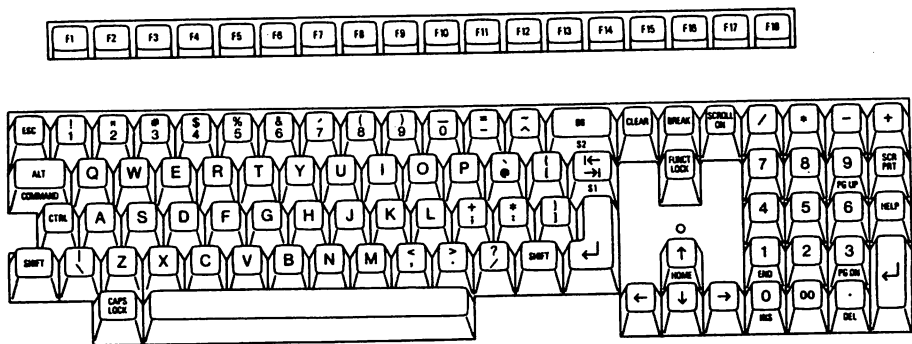
Figura 1.2

USA ASCII - TASTIERA 2

Figura 1.3

INTERROMPE LO SCROLLING SU VIDEO O L'USCITA SU STAMPANTE
SOSPENDE MOMENTANEAMENTE LO SCROLLING SU VIDEO
CTRL + H
= CANCELLA L'ULTIMO CARATTERE. IL CURSORE ARRETRA DI UNA POSIZIONE
TASTI FUNZIONALI
FISSA MAIUSCOLE
MAIUSCOLE E SIMBOLO SUPERIORE SUL TASTO
RETURN ENTER CR
CTRL
STAMPANTE ASSERVITA
SHIFT + PRTSC = HARD-COPY
CTRL + ALT
RIINIZIALIZZA IL SISTEMA

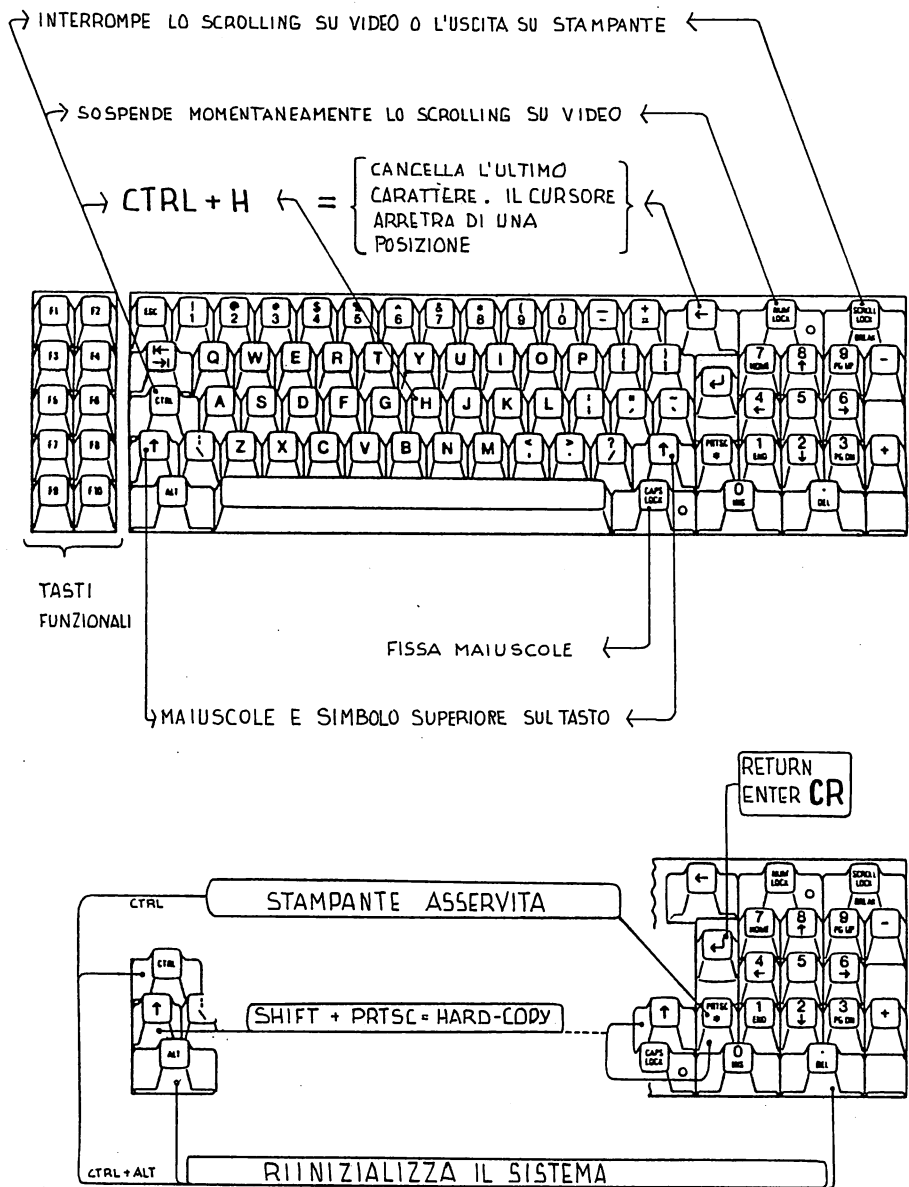



Figura 1.4

inglese e 21 con doppi simboli (segni d'interpunzione, le cifre numeriche e altre simboli speciali).

L'azione combinata col tasto ↑ (SHIFT), come abbiamo gia' visto, genera i caratteri maiuscoli delle lettere alfabetiche, mentre per i tasti con doppi simboli ottiene l'effetto di riprodurre quelli rappresentati nella parte alta del tasto stesso: contando anche lo spazio sappiamo, quindi, generare finora ben 2x(26+21)+1=95 caratteri grafici diversi.

Vediamo adesso l'impiego combinato degli stessi 47 tasti con i tasti speciali **CTRL** e **ALT**. In generale, se non sono state fatte manovre particolari, nessuno di essi provoca l'emissione di nuovi caratteri grafici, ma solo effetti speciali come la cancellazione di un carattere, il fissa maiuscole etc., oppure creano una particolare modificazione del comportamento della macchina, che gli addetti ai lavori amano definire come l'attivazione di una **funzione di sistema.**

A questa categoria appartengono delle funzioni standard, come l'interruzione forzata di un programma in esecuzione e altre funzioni stabilite direttamente dall'utilizzatore che di volta in volta le impiega nei suoi programmi, associandole liberamente a tasti scelti di suo gradimento.

Senza entrare troppo nel dettaglio, possiamo, quindi, limitarci a affermare per ora che tutti i tasti -salvo i tasti speciali ↑ (SHIFT), ALT e CONTROL- sono in grado di essere associati, alcuni in misura maggiore, altri in misura minore, a piu' di una funzione. In particolare nella figura 1.4, abbiamo suddiviso i tasti in due schiere: quella dei 47 tasti che, in combinazione con uno dei tre tasti speciali o da soli, possono generare ciascuno 4 diverse funzioni (grafiche e non), e quella dei 20 tasti che possono generare **solo tre** diverse funzioni, non avendo effetto quella in combinazione con il tasto speciale ALT che coincide con la funzione generata premendo il tasto singolo.

In totale, quindi, mediante la tastiera abbiamo la possibilita' di generare (47x4)+(20x3)=188+60=248 funzioni diverse: nella macchina con **tastiera USA ASCII**, con la quale stiamo lavorando, 95 di esse servono per stampare caratteri grafici (numeri lettere alfabetiche, simboli, etc.). Questi 95 caratteri grafici sono quelli che corrispondono ai caratteri stampabili della tabella ASCII (American Standard Code for Information Interchange). Con altra tastiera saranno possibili altri tipi di caratteri.

In questa tastiera abbiamo visto che a 153 tasti non sono stati assegnati compiti grafici. Cio' nondimeno ad essi sono state associate particolari funzioni di sistema, tra cui ricordiamo le tre piu' ricorrenti:

CTRL [SCROLL/LOCK]	interrompe momentaneamente l'attivita' che M24 ha in corso in quel momento;
CTRL [NUM/LOCK]	sospende momentaneamente lo scrolling su video, salvo riprenderlo alla pressione di un qualsiasi tasto;
CTRL [H]	cancella il carattere appena introdotto e visualizzato sul video (lo stesso effetto puo' ottenersi col tasto dedicato [⟵])

A due delle suddette funzioni di sistema segue normalmente un'azione di ritorno alla situazione di partenza. Cosi' per riprendere le attivita' interrotte dopo aver impostato CTRL SCROLL/LOCK basta introdurre la parola CONT (continua); per la funzione CTRL H, invece, non c'e' altro da dire, salvo che, cancellato un carattere, non si puo' piu' recuperarlo.

Un'altra importante funzione di sistema e' il fissa maiuscole: la si ottiene mediante il tasto apposito [CAPS LOCK]. Dopo aver premuto questo tasto si illumina la lampadina spia in esso incorporata a ricordarci la funzione attivata e da quel momento tutti i tasti alfabetici provocheranno la stampa di caratteri in maiuscolo, mentre non si avra' alcun effetto sui tasti con doppio simbolo. Durante questo blocco delle maiuscole, se desideriamo riottenere qualche carattere minuscolo basta premere il tasto in accoppiata con lo SHIFT (la situazione si e' quindi invertita rispetto a quella in cui l'assetto normale era dedicato alle minuscole).

Per sbloccare le maiuscole e' sufficiente premere ancora il tasto [CAPS LOCK]. Ovviamente per i tasti con doppio simbolo, in cui la combinazione fissa maiuscole non ha effetto, occorre sempre premere lo SHIFT per ottenere il simbolo impresso nella parte alta del tasto.

Un'avvertenza: non premere a lungo il tasto, se non desiderate la ripetizione automatica del simbolo.

Nella figura 1.4 sono illustrate tutte le funzioni di tastiera fin qui descritte.

Trattiamo infine uno degli effetti piu' importanti, finora appena menzionato, cioe' quello associato alla consegna interna all'unita' centrale dei dati introdotti. Questa consegna avviene premendo il tasto speciale contraddistinto dal simbolo ↵ (una L rovesciata con freccia verso sinistra). Per esigenze di semplicita' tipografiche in questa sede designeremo il tasto ↵ come [CR] dalle iniziali delle parole CARRIAGE RETURN (ritorno carrello). La scelta di tale nome deriva dall'effetto che provoca subito dopo averlo premuto. Vediamo di che cosa si tratta.

Abbiamo detto che i dati introdotti, oltre che essere visualizzati sullo schermo, sono anche accumulati in un deposito di tastiera per consentire di compensare la modesta velocita' di battitura manuale del testo rispetto a quella di cattura ed elaborazione dei dati da parte della macchina. La quantita' di caratteri introdotti in questo deposito (chiamato buffer di tastiera), che ha una capacita' di 128 caratteri, non corrisponde pero' a quella che puo' apparire dal video, che ne visualizza molti di piu'.

Cosi' se l'operatore non scarica all'unita' centrale (premendo il tasto [CR] il contenuto del buffer prima che questo raggiunga il limite dei 128 caratteri e supera tale capacita', gli ulteriori caratteri visualizzati sullo schermo non hanno alcuna possibilita' di essere consegnati all'unita' centrale: alla pressione del tasto [CR], i caratteri eccedenti vanno irrimediabilmente perduti. Viceversa, finche' tale limite non viene raggiunto, sta nella facolta' dell'operatore di premere o no il tasto [CR], mentre M24 rimane pazientemente in attesa.

1.2.3 IL VIDEO

A questa periferica fondamentale dovremo dedicare particolare attenzione, sia perche' dai suoi attributi visivi dipende l'effetto grafico con cui appaiono i risultati, sia perche' nella fase di introduzione dei dati da tastiera, consente l'eventuale correzione di quelli errati. Questa interazione riguarda anche gli effetti speciali creati dalla pressione di particolari tasti.

Esaminiamo intanto quanti caratteri puo' contenere il video di M24. Tale numero dipende da una selezione operata a inizio lavori, un po' come in un motore che puo' funzionare a gas e a benzina. Quando si sceglie la massima capacita' si possono visualizzare fino a 2000 caratteri su 25 righe di 80 caratteri ciascuna.

E' possibile selezionare densita' minori, ottenendo pero', in contropartita, caratteri piu' grandi. Ad esempio a bassa densita' la capacita' complessiva scende a 1000 caratteri distribuiti su 25 righe di 40 caratteri ciascuna.

Con questa densita', per quanto riguarda la generazione dei caratteri, esaminando solo gli aspetti geometrici e tipografici, ognuno di essi e' formato accendendo elettronicamente i punti di incrocio di 7 linee verticali con 13 orizzontali.

In tale gabbia 7x13, in effetti, la selezione dei punti da accendere cambia da carattere a carattere (vedi la figura 1.5) per tener conto della diversa estensione verso l'alto o il basso della gabbia.

La dislocazione dei caratteri sullo schermo avviene come la scrittura manuale per gli occidentali, cioe'automaticamente da sinistra verso destra, una riga sotto l'altra, secondo un particolare reticolo.

Immaginiamo lo spazio utile del video (215 mm x 161.25 mm) solcato da 640 linee verticali a distanza di circa 3/10 di mm e di 400 linee orizzontali a distanza di 4/10 di mm.

Per motivi di intelligibilita', nella generazione dei caratteri, di tali solchi grafici ne viene utilizzata solo una parte, lasciando un corridoio tra riga e riga e un piccolo intervallo tra carattere e carattere.

Ovviamente scegliendo la densita' di 25 righe da 80 caratteri, anziche' quella da 25 righe da 40 caratteri, lo spazio tra le righe e' piu' ampio, in quanto i caratteri sono piu' grandi.

Piu' precisamente -si veda anche la figura 1.5- sono di un solco orizzontale e tre verticali nella densita' maggiore e di un solco verticale e senza alcun solco orizzontale nelle altre due densita' minori.

Abbiamo cosi' introdotto la nozione di punto luminoso che chiameremo pixel -dalla fusione delle parole inglesi picture element (elemento grafico).

Sul video M24 di pixel se ne possono accendere 256000 (640x400) e tale disponibilita' sara' completa -come vedremo al capitolo 4- quando lo utilizzeremo in modo grafico, mentre per la scrittura, come abbiamo appena dimostrato, per esigenze pratiche dei singoli alfabeti, se ne sfrutta solo la parte destinata a localizzare i vari caratteri del testo.

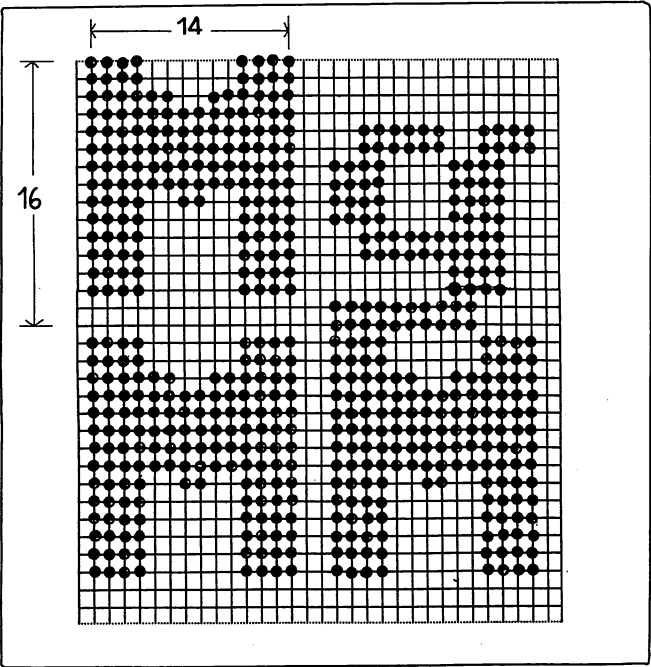
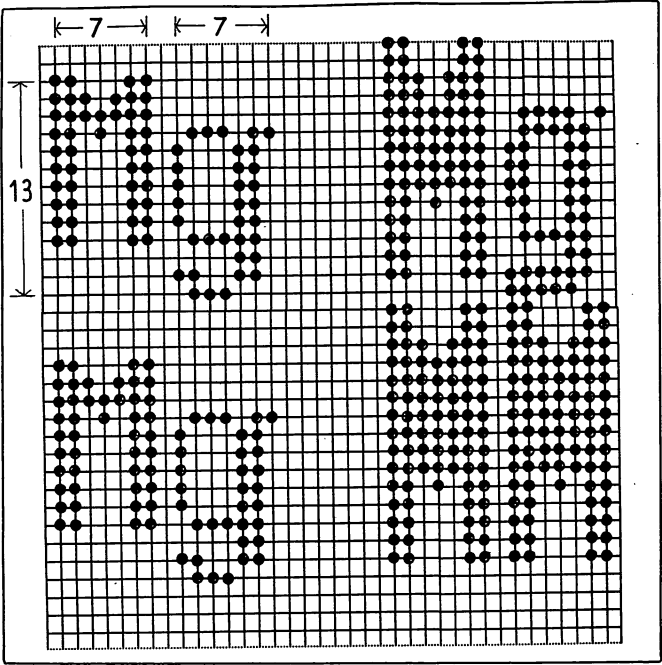



Figura 1.5

La luminosita' dei pixel viene regolata da una manopola sul retro del video, in alto a destra. Per particolari effetti ottici (e per combattere la noia o l'affaticamento della lettura su video) e' possibile, sempre via software, invertire il colore di fondo con quello di scrittura. In un video monocromatico significa poter scrivere bianco su fondo nero (situazione normalmente offerta dalla macchina), oppure nero su fondo bianco, come su una pagina di carta. E', inoltre possibile selezionare fino a 4 tonalita' di grigio.

In merito al colore esistono due versioni di video: quello monocromatico e quello con quattro colori selezionabile su una tavolozza di 16. Quando sia presente una particolare estensione elettronica e' possibile disporre di tutti e 16 i colori. Meccanicamente, il video e' completamente indipendente e puo' quindi essere appoggiato sopra l'unita' base, o posto lateralmente.

1.2.4 I DISCHETTI

Dischetto e' l'equivalente italiano del termine inglese **floppy disk** e costituisce la memoria ausiliaria standard di cui M24 e' dotato. A questa memoria periferica si fa ricorso per piu' motivi: ad esempio quando le informazioni non stanno piu'dentro la capacita' di quella centrale. Inoltre, per motivi di conservazione delle informazioni, tali tipi di supporti sono l'unica sede non labile di notevole capacita' di cui il sistema e' dotato per la conservazione delle informazioni. Infatti, come sappiamo, il contenuto della memoria principale viene perduto nel momento in cui la macchina viene spenta.

Grazie a questa ingente capacita' di memoria ausiliaria e' possibile archiviare interi testi e richiamarli in qualsiasi istante sul video senza alcuna necessita' di utilizzare carta e riducendo al minimo l'ingombro dell'archiviazione: basti pensare che su un dischetto da 360 kB puo' stare una quantita' di caratteri pari al contenuto complessivo di ben 180 cartelle dattiloscritte.

Passiamo ora in rassegna le caratteristiche fisiche dei dischetti. Si presentano (vedi figura 1.6) come una busta quadrata di cartoncino nero di circa 13 cm di lato e con un foro centrale di quasi 3 cm di diametro.

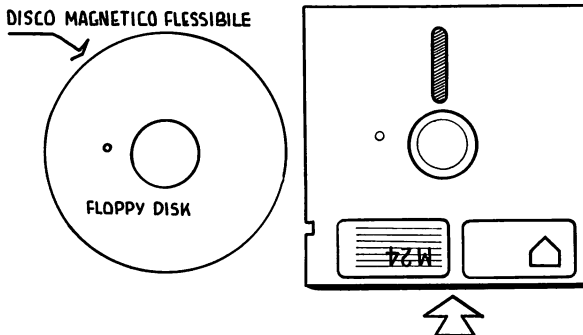



Figura 1.6

La busta contiene un disco di plastica ricoperto di ossido di ferro come il nastro magnetico per musicassette. Tale disco posto in rotazione, offre la sua superfice magnetizzabile attraverso un foro ovale dove puo' essere tastato da una testina magnetica.

Il solco magnetico non e' a spirale come nei dischi musicali, ma si chiude su se stesso a circonferenza: in tal modo ad ogni posizione assunta dalla testina, in virtu' del movimento radiale della stessa, corrisponde un solco ideale, o **pista** (tracciata dal disco in movimento) in ciascuna delle quali possiamo registrare informazioni e rileggerle in tempi successivi.

Senza entrare in troppi dettagli, limitiamoci a conteggiare quante informazioni stanno su un dischetto. Il risultato di questo calcolo dipende da almeno tre fattori: la densita' radiale delle piste, la densita' delle informazioni lungo la pista circolare e il numero delle facce utili del dischetto (1 o 2).

Nei dischetti da noi utilizzati, aventi una capacita' nominale di 360 kB (360x1024=362496 bytes), la struttura e' la seguente:

- 2 facce registrabili
- 48 piste per faccia
- 9 settori per pista

In effetti per motivi tecnici, la capacita' realmente utilizzabile e' di 322560 bytes.

Un'altra caratteristica dei dischetti e' la loro velocita' di rotazione (300 giri al minuto), che influenza il tempo medio di accesso. Rispetto ai nastri magnetici, il grande vantaggio dei dischi e' la possibilita' di accedere al settore semplicemente portando la testina sulla pista in cui si trova tale settore e, nel peggiore dei casi - se il settore e' appena passato- aspettare un giro del disco (0.2 secondi) perche' si ripresenti sotto la testina magnetica.

L'accesso e' quindi molto piu' diretto (viene impropriamente detto casuale) di quello forzatamente sequenziale del nastro magnetico.Il trasferimento avviene a 250000 bits al secondo (circa 30000 caratteri/s).

Ovviamente i dischetti devono essere inseriti nel meccanismo frontale di M24, all'interno del quale vengono fatti ruotare e, a seguito di particolari comandi, sono toccati e letti dalle testine. Il complesso dei meccanismi che realizza l' unita' periferica delegata a svolgere tutte queste funzioni asservite alla CPU si chiama **drive.** A questo riguardo esistono due versioni di M24: quella con un solo driver e quella con due.

1.2.5 LA STAMPANTE

Descriviamo ora le caratteristiche di una delle tante stampanti collegabili a M24: la PR 15-B (OPE DM 5060). Questa unita', che normalmente fa parte della configurazione standard di riferimento, e' una stampante da tavolo da 8 kg che stampa alla velocita' di 120 caratteri al secondo. Ogni carattere viene stampato per punti secondo gli incroci ideali di 7 righe verticali e 9 orizzontali.

La lunghezza della riga di stampa puo' avere due densita' selezionabili da microinterruttori posti nella parte posteriore della macchina: la densita' piu' alta di 16.6 car/pollice equivalente a un passo di scrittura di 1.53 mm, consente di stampare righe di 132 caratteri, mentre con densita' di 10 car/pollice la riga ne puo' contenere solo 80. Sono possibili anche densita' di 8 caratteri per pollice (tipo Elite) e varie combinazioni di caratteri allargati con diversi attributi grafici (grassetto, doppia battuta, di qualita', etc.).Si veda al riguardo il paragrafo 10.3.

Per quanto riguarda l'alimentazione dei moduli di carta, questa puo' avvenire in modo manuale (un foglio per volta), oppure in automatico continuo con trascinamento della carta mediante fori laterali. La stampante PR 15-B oltre alla stampa dei caratteri di un testo in modo tradizionale, puo' riprodurre fedelmente per punti il contenuto del video: si ottiene in tal modo l' **hard-copy** del video.

1.3 COME FUNZIONA M24

Dopo avere passato in rassegna le principali caratteristiche di una normale configurazione M24 -quella alla quale faremo costante riferimento in tutto il libro- vediamo di mettere a fuoco gli aspetti pratici del funzionamento della macchina, spiegando le modalita' per attivarla e farla operare per obiettivo. Nei successivi paragrafi metteremo quindi insieme quanto gia' sappiamo di M24 per acquisire schemi pratici di funzionamento.

1.3.1 IL SISTEMA OPERATIVO

Un sistema operativo e' un insieme di procedure automatiche per gestire il rapporto tra le richieste dell'utilizzatore e le risorse della macchina. Talvolta, si usa parlare di **ambiente operativo** anziche' di sistema operativo, intendendo cosi' le condizioni in cui vengono posti la macchina e il suo interlocutore per compiere un certo lavoro

Per meglio comprendere le funzioni di un sistema operativo e il perche' lo si chiami anche ambiente , riferiamoci ad esempi semplici tratti per analogia da casi che possiamo incontrare nella vita di tutti i giorni.

La cucina e le **ricette** sono esempi consunti ma ottimi per la loro analogia con un sistema operativo e i **programmi applicativi**.

Nella cucina formano l'ambiente: il forno, i fuochi, le batterie dei recipienti e gli ingredienti alimentari non ancora elaborati.

Un altro esempio di ambiente e' quello per la lavorazione del legno. Esso e' costituito, oltre che da un banco e dal legno, anche da tutti gli attrezzi necessari per eseguire le lavorazioni, compreso il trapano con le relative punte.

In entrambi gli esempi l'ambiente e' costituito da un insieme di risorse (strumenti e componenti per la lavorazione) che in un certo senso condizionano il risultato.

Dovendo lavorare in quegli ambienti per raggiungere un certo obiettivo, puo' accadere di non aver tutto il necessario: materia prima, attrezzi, etc. In tal caso, prima di rinunciare, si cercano generalmente soluzioni alternative, che al limite possono anche costringere a rivedere lo stesso prodotto della lavorazione. In tema di attrezzi, ad esempio, se non ho il trapano elettrico, posso usare il trapano a mano. Difficilmente, pero', dovendo fare un foro, potro' fare a meno del trapano.

In definitiva, mediante un sistema operativo, e' possibile impiegare l'elaboratore al meglio delle sue risorse fisiche (capacita' di memoria, velocita' di elaborazione, etc.) liberando l'utilizzatore di tutte le operazioni o procedure accessorie intermedie.

Senza i servizi di utilita' messi a disposizione dal sistema operativo l'ambiente di impiego di un elaboratore diverrebbe allucinante e il lavoro di preparazione dei programmi applicativi, "il far cucina", un lavoro estenuante e di altissima specializzazione.

A questo punto, con gli esempi fatti, qualcuno potrebbe aver confuso gli utensili di laboratorio con le parti materiali dell'elaboratore.

Possiamo fare allora un ulteriore passo di avvicinamento alla comprensione di un sistema operativo, pensando alle operazioni necessarie per registrare su una cassetta musicale un brano letto da un disco. Le operazioni da compiere in questo caso sono almeno di due tipi:

Fisico, per la predisposizione dei supporti. Il disco deve esere messo sul piatto dalla parte dove c'e' il brano. Le apparecchiature di collegamento tra registratore e giradischi devono essere regolate correttamente. A brano terminato, occorre fermare gli apparecchi.

Logico. [a]. La cassetta e' vergine. La durata del brano deve essere non superiore alla capacita' della cassetta, altrimenti non riusciamo a registrarlo completamente. [b]. La cassetta non e' vergine e il brano deve essere registrato al posto di un'altro che non interessa piu' conservare.

In quest'ultimo caso la situazione e' piu' complessa perche' bisogna fare una verifica di tutti gli spazi resi disponibili, corrispondenti a brani cancellabili, e scegliere quello la cui durata piu' si avvicina a quella del brano da registrare. Naturalmente per far cio', se non si tiene una contabilita' degli spazi liberabili e della loro esatta dislocazione nella cassetta, occorrera':

- far svolgere il nastro della cassetta
- annotare gli inizi e le fini di ogni brano e insieme tener conto della loro durata,
- decidere quali possono essere sacrificati a favore di altri.

Solo dopo queste operazioni si ricade nel caso [a].

E ora potremmo chiederci: "Ma vale veramente la pena far tutto cio'?"

La sensazione che proviamo nel trattare questo tipo di problemi e' a mezza strada tra lo sgomento e la noia: sicche' quasi nessuno attua questa specie di gestione dei brani musicali, con la conseguenza che ogni nastroteca contiene un'alta percentuale di brani indesiderati.

Immaginate ora un registratore che esegua automaticamente tutti quei passi e che all'inizio di un'operazione di scrittura di un brano (da disco, da radio, da tv, dal vivo) vi chieda con quale nome desideriate archiviarla e la sua presumibile durata. Non solo, ma nel momento del recupero dello spazio di un brano da sacrificare, vi proponga l'elenco di tutti i pezzi registrati su quella cassetta, con la relativa durata.

Ebbene, questo tipo di struttura operativa potremmo chiamarla "il sistema operativo" a bordo del registratore XY: infatti, cio' che quella struttura realizza e' una serie di servizi di utilita' per l'operatore.

Il messaggio che possiamo trarre dall'esempio appena trattato e' che il lavoro assistito da un certo ambiente predisposto e attrezzato a guidare e snellire il lavoro stesso, diventa non solo piu' facile, ma anche piu' sicuro perche' esiste una procedura organizzata e automatica che ne supervisiona le parti piu' ripetitive e noiose, lasciando libere le sole funzioni esterne di scelta e alcune notizie che il sistema ignora: nel nostro caso i nomi del nuovo brano, con relativa durata, e di quello da cancellare.

Secondo quanto abbiamo finora osservato, possiamo ora cominciare a delineare i compiti principali del sistema operativo di un elaboratore. Fate attenzione che, anche se vestito del piu' potente sistema operativo e quindi piu' docile e semplice da usare rispetto a una macchina costituita di solo hardware, l'elaboratore non e' ancora capace di far qualcosa di utile per noi.

Per questo un sistema operativo si chiama anche **software di base**: una base sulla quale impiantare un programma applicativo.

Con l'aggiunta di un programma specifico per fare un determinato lavoro, la macchina e' finalmente pronta ad accettare i comandi definitivi che la rendono operante e capace di effettuare le elaborazioni sui dati che le verranno comunicati dall'operatore e produrre cosi' i risultati richiesti.

L'insieme del programma applicativo e del sistema operativo e', in senso lato, il software di una macchina.

Uno dei sistemi operativi del quale M24 puo' vestirsi e' l'MS DOS (MicroSoft Disk Operating System; altri sono il **C/PM**, l'**UCSD** e il PCOS. Quest'ultimo e' l'ambiente nativo dell'M20 e puo' essere riprodotto in M24 con una piastra di elettronica addizionale (APB).

Non potendoli trattare tutti in questa sede, per le inevitabili differenze sul piano pratico, ci limiteremo al primo che, tra l'altro, e' quello per il quale la macchina ha raggiunto la maggiore diffusione. Il primo elemento software agganciato quando M24 viene acceso e' il nucleo dell'MS-DOS che viene estratto dal dischetto, che per questo viene chiamato **dischetto sistema**, per mezzo del caricatore hardware della macchina. Dal nucleo si irradia un ambiente operativo con il quale e' possibile **attrezzare** alcune capacita' della macchina, ma non tutte: le altre risiedono sul dischetto sistema e per farle entrare nell'operativita' della macchina occorre richiamarle espressamente, attraverso alcuni **comandi** compresi nel nucleo.

Caricarle tutte sarebbe come -tornando per un'attimo in quel laboratorio per la lavorazione del legno- attrezzare la totalita' delle macchine utensili disponibili, anziche' solo quelle che servono per una certa fase di lavoro. Per maggiori dettagli si consulti l'Appendice B.

1.3.2 I MODI OPERATIVI

La figura 1.7 mostra una serie di attivita' iniziate dall'operatore a partire dall'accensione della macchina. Si suppone naturalmente di aver introdotto il disco sistema contenente MS-DOS nel drive inferiore (denominato A) di M24.

All'accensione della macchina, dopo una fase automatica di autodiagnosi, avviene il caricamente in memoria principale del nucleo dell'MS-DOS. Della effettiva disponibilita' di questo nucleo si ha nozione allorquando vediamo lampeggiare una lineetta fatta di microscopici punti luminosi, subito dopo il simbolo ">".

Questa lineetta, che viene chiamata **cursore**, ci avverte che il sistema e' in attesa di comandi. L'operatore puo' approfittarne -fase 2 della figura- per chiedere ad esempio il caricamento in memoria di uno dei moduli del sistema operativo. Tra questi, l'interprete del **BASIC**, che su M24 si chiama GWBASIC, una delle piu' potenti versioni di Basic esistenti, dotata di tutte le piu' sofisticate semplificazioni in tema di grafica avanzata e di gestione dei suoni e dei colori.

Il comando per chiamare questo Basic viene comunicato all'MS-DOS battendo il nome GWBASIC fatto seguire dal messaggio di chiusura generato dalla pressione del tasto [CR].

Una volta eseguito questo comando, la memoria centrale viene in parte invasa da questo componente che ha la funzione specifica di interpretare le istruzioni scritte in Basic, insieme con alcuni servizi di utilita' connessi con tale linguaggio. Di questo argomento tratteremo al prossimo paragrafo e dal capitolo 2 in poi: per adesso rileviamo un altro particolare importante. Nel momento in cui l'MS-DOS ha portato in memoria centrale il GWBASIC sul video e' cambiato il simbolo che precede il cursore: al posto del simbolo ">" e' apparsa la dicitura [**Ok**].

Si tratta del segnale di riconoscimento del Basic che ci avverte della disponibilita' immediata di tale linguaggio. Questo componente del sistema operativo, normalmente archiviato sul dischetto sistema, e' stato risvegliato e portato in memoria principale, come strumento di lavoro. L'ambiente M24 si e' quindi arricchito di questo basilare attrezzo informatico, grazie all'MS-DOS che lo ha attivato. E' pero' cambiato anche il modo di trattare la macchina, che da quando ha il Basic in memoria ha un nuovo modo di comportarsi e di reagire.

Di tale nuovo modo operativo bisogna ovviamente rispettare le regole, con la massima precisione. Tra queste c'e', naturalmente, quella che ordina a M24 di ritornare al modo MS-DOS Per impartire tale comando occorre battere la parola "**system**" e ricomparira' il segno ">", e insieme perderemo la disponibilita' del Basic in memoria.

ACCENSIONE M24
ON
INSERIMENTO DISCHETTO E CARICAMENTO MS DOS
SEGNALE DI RICONOSCIMENTO DI MS-DOS
>
DISCHETTO SISTEMA
FASE 1
FASE 2
RICHIESTA INTERPRETE GW BASIC
SEGNALE DI RICONOSCIMENTO DEL BASIC
OK
Gwbasic [CR]
RICHIESTA PROGRAMMA APPLICATIVO
load [CR]
INGRESSO DATI
TASTIERA
[CR]
ELABORAZIONE
USCITA RISULTATI
VIDEO

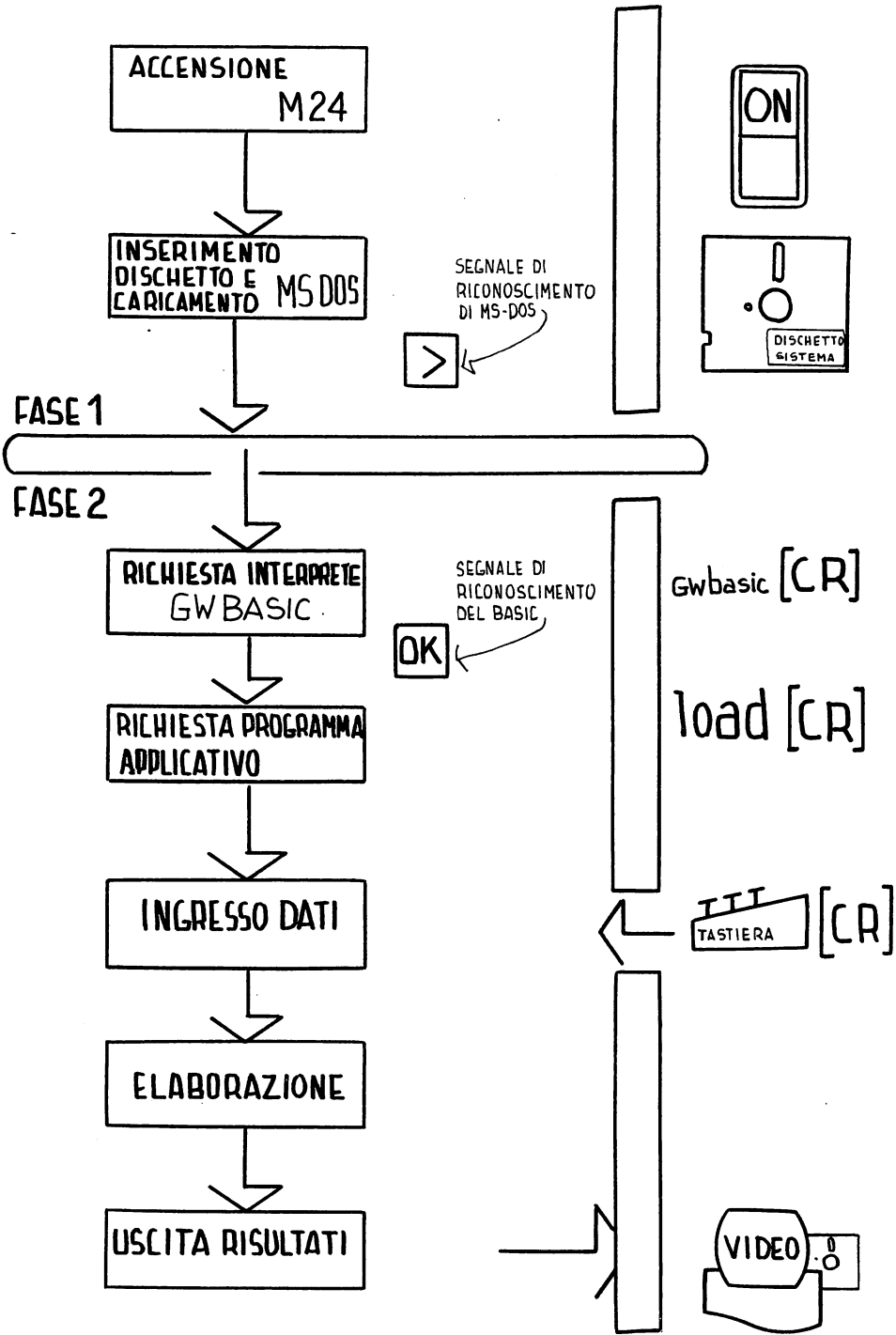



Figura 1.7

Ricordarsi in ogni istante in qual modo M24 stia operando e' quindi essenziale per evitare spiacevoli sorprese: l'ambiente MS-DOS e' dotato di certe risorse tra cui l'abilita' di farci passare in ambiente Basic. Potreste chiedervi se non sarebbe stato piu' opportuno aver sempre tutto a disposizione, anziche' proccuparsi ogni volta di portarsi le cose strettamente necessarie. La risposta e' ancora una volta di natura pratica e economica: sarebbe come se partendo per una vacanza al mare portassimo in valigia un passamontagna...perche' non si sa mai...

Evitiamo allora di dare comandi alla macchina che non siano pertinenti con il modo in cui sta operando in quel momento, se non vogliamo risposte contenenti messaggi che ci segnalano situazioni di errore e di ambiguita'. Perche' le cose vadano bene dobbiamo continuamente ricorrere alla segnaletica di macchina: a tal scopo servono i segnali di riconoscimento, che, ripetiamo, sono ">" per MS-DOS e "**Ok**"per il Basic. Nella figura 1.8 sono illustrate le fasi appena trattate.

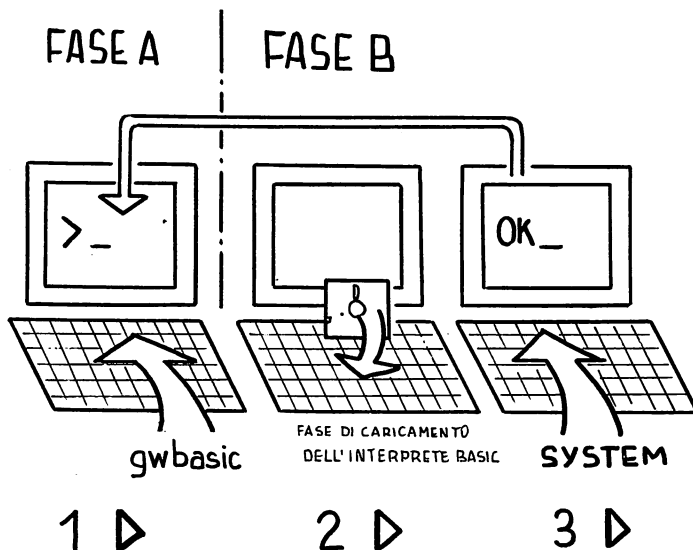



Figura 1.8

Per completare l'analisi del diagramma , una volta che il Basic e' stato caricato in memoria mediante i suoi specifici comandi, possiamo chiamare da dischetto il particolare programma di elaborazione che ci interessa al quale affidare i dati di partenza che introdurremo da tastiera. A tale fase segue il trattamento e, infine, la produzione dei risultati nella forma prevista dal programma e sulle unita' periferiche assegnate (video e/o stampante).

Torniamo ora un po' alla pratica operativa sul sistema, anche se in parte dovremo ripeterci. Accendiamo il sistema M24. Se vogliamo impiegare la stampante accendiamo anche quella, curando che sia collegata alla rete e all'unita' base di M24, e abbia almeno un foglio di carta inserito e in presa col rullo.

Per maggiori informazioni sulle stampanti e' stato dedicato l'intero paragrafo 10.3.

Montiamo quindi il dischetto sistema (che contiene l'MS-DOS) nel drive A (quello inferiore). Se volete e' possibile montare il disco sistema nel drive B (quello superiore), anziche' in A, ma, allora, bisogna far si' che la lineetta del cursore lampeggi dopo il simbol "B>": cio' si ottiene introducendo: [B:] [CR].

In ogni caso dopo aver introdotto il dischetto in una delle due fessure sul frontale di M24, dalle quali si accede ai drive, occorre debitamente richiudere il relativo sportello.
Particolare cura dovra' aversi nel maneggiare i dischetti sia nei confronti delle superfici magnetiche esposte (non toccarle in quei punti) sia nella fase di introduzione nel driver. Tale operazione va fatta con delicatezza, con quel tanto di pressione sufficiente a far raggiungere una posizione in cui il dischetto e' completamente inserito e si sia avvertito un "clic" di aggancio. A questo punto potete chiudere lo sportello del drive.

Anche in questa operazione, se notate qualche anomalia o intoppo, non forzate mai il dischetto, ma risfilatelo e ripetete l'intera operazione da capo. Fate inoltre la massima attenzione che, durante la manovra di inserimento dei dischetti, non sia mai accesa la lampadina rossa del drive. Potrebbe essersi accesa perche' avete inavvertitamente premuto il tasto [CR] e inviato un messaggio al quale l'MS-DOS tenta di dar risposta, ad esempio prelevando qualcosa proprio dal drive vuoto nel quale state inserendo un dischetto. Un'operazione maldestra di introduzione o estrazione del dischetto dal drive a spia accesa puo', in generale, provocare seri danni al dischetto e al drive.

Prendiamo quindi l'abitudine di accendere M24 solo **dopo** aver selezionato i dischetti con cui vogliamo lavorare e averli accuratamente introdotti nei rispettivi drive.

Se tutto e' in ordine e si accende -vedi figura 1.9- la spia rossa sulla tastiera di M24, dovreste notare una serie avvertimenti sul risveglio della macchina e l'emissione di un "bip" da parte dell'avvisatore acustico.

Piu' o meno contemporaneamente sentirete un lieve rumore di motore che gira (e' quello del drive per il trascinamento del dischetto), mentre si accendono alternativamente le spie luminose della tastiera e quella del drive. Se avete dimenticato di inserire il dischetto dell'MS-DOS nel drive, comparira' sul video un messaggio lampeggiante che vi invitera' a farlo.

Il messaggio e' purtroppo in inglese, per le note ragioni di internazionalita' del prodotto, e termina con le parole "....Primary Boot Strap...DISK READ ERROR", che significa..."Errore di lettura disco...". Comunque sia, il primo messaggio costruttivo che appare e':

Microsoft MS-DOS versione 2.11...

Infatti subito dopo appare la prima videata, cioe' il primo messaggio sostanziale per l'utilizzatore (vedi figura 1.10).

Figura 1.9

```
Resident Diagnostics
Rev 1.1

CPU (i8086)   Pass
ROM Module    Pass
DMA Timer     Pass
DMA Control   Pass
Interrupts    Pass
256 kb RAM    Pass
RT Clock      Pass
Fixed Disk    Not Present
Floppy (A:)   Not Ready

Select Alternate CPU (y/n)?n

Primary Boot-Strap...

Microsoft MS-DOS version 2.11
Copyright 1981,82,83 Microsoft Corp.
Rev. 1.01  6/29/84

Command v. 2.11
Current date is Wed  1-16-1985
Enter new date:
```

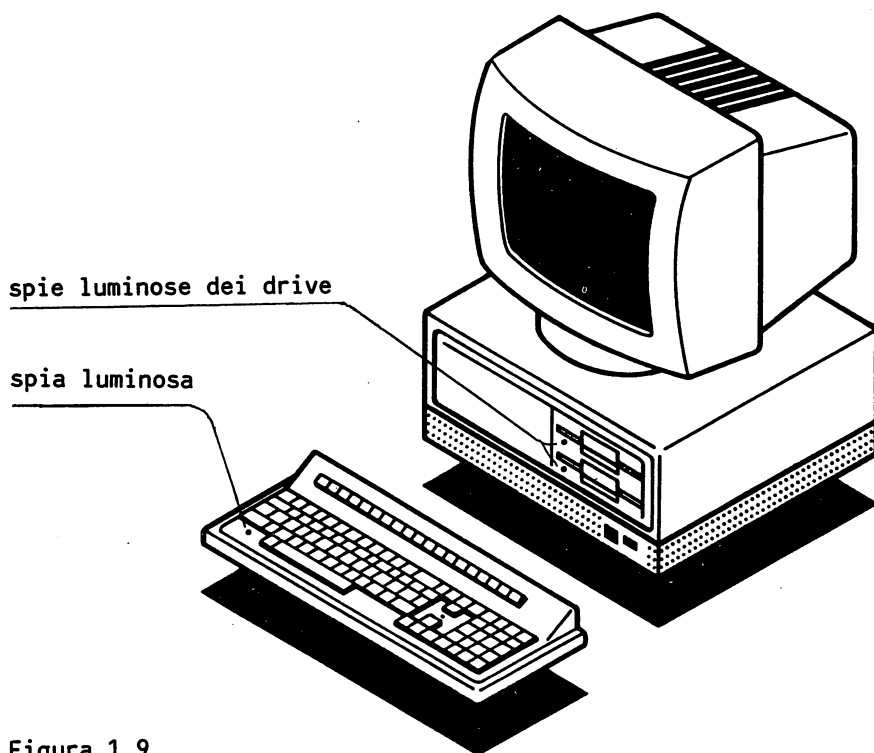
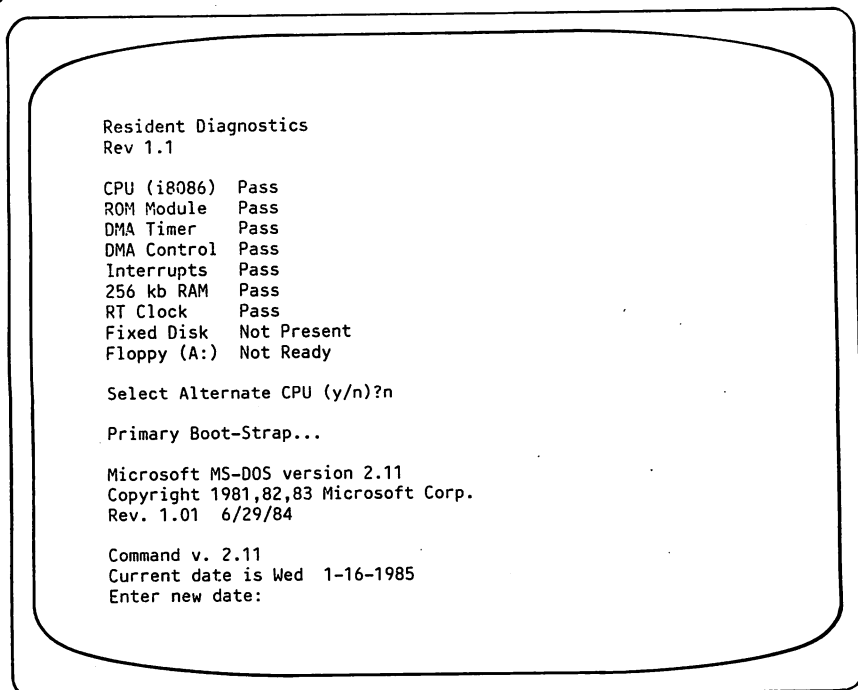



Figura 1.10

Esso contiene informazioni sulle caratteristiche hardware e software della macchina.

La cosa che salta subito all'occhio, a fine videata, e' il cursore luminoso, quella lineetta lampeggiante, di cui abbiamo gia' trattato, che segue il segnale di riconoscimento dell'MS-DOS.

Una volta apparso questo segnale, M24 si pone in attesa di ordini. Chiaramente, non possono essere ordini improvvisati o di fantasia. Occorre conoscere lo strumento e adoperarlo nel modo corretto. Qualunque azione che non sia meditata, nel senso che non ne abbiate previsto l'effetto, puo' portarvi in una zona inesplorata di M24, quella zona di cui non sapete ancora niente e, quello che e' peggio, nella quale vi trovereste intrappolati senza via di uscita, a meno di non spegnere la macchina e ricominciare tutto da capo.

Una volta che abbiate deciso di correre questo rischio, date pure un ordine qualunque al PCOS, e battete una lettera a caso, ad esempio la lettera C. Il carattere corrispondente appare sul video esattamente dove prima era il cursore lampeggiante, mentre quest'ultimo guadagna la posizione a destra immediatamente adiacente.

Comincia cosi' a chiarirsi la funzione del cursore: esso indica il punto del video in cui andra' a finire il carattere che state per digitare.

Orbene, se tutto il vostro messaggio e' costituito dalla sola lettera C, vorrete anche sapere come farlo arrivare fino a M24. Infatti, dopo aver digitato e fatto apparire sul video un carattere, non succede assolutamente niente, finche' non battete il tasto [CR]. Allora la manovra produce una risposta immediata sul video: Bad comman or file name. (La macchina non capisce e avanza ipotesi di errore:...avete, forse introdotto un falso comando o il nome sbagliato di un file?).

Digitando un numero seguito dal tasto [CR] la risposta sara' identica.

E ora, prima di passare all'analisi di qualche comando MS-DOS, facciamo un po' di pratica sulla tastiera.

Tasto **[↑]** o **SHIFT.** E' il fissa maiuscole della macchina da scrivere con un po' d'intelligenza e qualche complicazione in piu'. Quando viene premuto simultaneamente ai tasti alfabetici provoca l'introduzione delle lettere maiuscole corrispondenti; se utilizzato, invece, con tasti a doppio simbolo produce l'immissione del carattere posto nella parte alta del cappuccio.

Si puo' sperimentare l'effetto dello SHIFT anche sulla tastiera numerica: la digitazione dei tasti che portano solo i numeri o gli operatori aritmetici (* / + -), mentre teniamo premuto il tasto SHIFT, ha per effetto di annullare ogni introduzione.

Tasto **CAPS LOCK.** Premendo questo tasto e' possibile mantenere l'effetto maiuscolo sulle lettere dell'alfabeto. La cosa non vale per i tasti con doppio simbolo per i quali bisogna ancora usare il tasto SHIFT. Si noti che durante la battitura di un testo in modo maiuscolo se vogliamo introdurre qualche carattere minuscolo useremo ancora lo SHIFT: l'effetto e' quindi complementare a quello normale. Per tornare infine alla situazione di partenza e sbloccare le maiuscole,

useremo la stessa tecnica con la quale vi siamo entrati. Lo stato di attivita' del tasto CAPS LOCK appare dalla lampadina incorporata nel tasto stesso.

Tasto **NUM LOCK.** Anche questo tasto provoca un'alterazione del funzionamento di alcuni tasti: si tratta di quelli della sezione destra destinati normalmente (nella tastiera internazionale tipo 1) al governo del cursore.

Premendo, invece, NUM LOCK si attivano, con gli stessi tasti i valori numerici da 0 a 9 scolpiti nella parte superiore del cappuccio.

Tasto [←]. Ogni volta che si batte questo tasto si effettua la cancellazione dell'ultimo carattere immesso. La stessa operazione si effettua battendo il tasto H, tenendo premuto CTRL.

Tasto **CTRL** (control). L'accoppiata con questo tasto svolge molte funzioni importanti. Tre di queste vanno acquisite al piu' presto: la prima funzione riguarda l'interruzione forzata di qualsiasi operazione la macchina abbia in corso in quel momento. Tale interruzione si applica battendo il tasto SCROLL LOCK (che per questo ha scolpito la parola BREAK sul fianco anteriore) mentre si tiene premuto il tasto CTRL. La seconda funzione riguarda una sospensione dello scorrimento (scrolling) del video. Tale sospensione si applica battendo il tasto NUM LOCK (che si trova a sinistra di SCROLL LOCK) mentre si tiene premuto CTRL; qualsiasi tasto fa riprendere l'azione sospesa.

La terza funzione riguarda l'asservimento della stampante collegata: battendo il tasto PRT SCR (che sta sotto il tasto [CR]), mentre mantenete premuto CTRL agganciate la stampante collegata all'M24, che, naturalmente, deve essere accesa e alimentata con un foglio di carta: da quel momento qualunque cosa introduciate da tastiera viene anche stampata. Per mettere al silenzio la stampante basta ripetere la stessa operazione che l'aveva attivata.

<u>L'impostazione dei parametri di stampa.</u> Per gli aspetti tipografici c'e' un particolare comando MS-DOS che ha per scopo la modifica dei valori dei parametri di stampa rispetto a quelli che la macchina assegna automaticamente in assenza di diverse disposizioni (i parametri automatici si chiamano di **default**).

Digitate il messaggio mode **lpt1:132,8** [CR] e farete passare la stampante PR 15-B collegata, dalla densita' di 10 caratteri per pollice (righe da 80 caratteri) con interlinea 6 righe per pollice a quella di 16.6 caratteri per pollice (righe da 132 caratteri) con interlinea 8 caratteri per pollice.

Naturalmente per tornare alla densita' di partenza, bastera' scrivere l'ordine:**mode lpt1:80,6.** Avrete certamente notato i vari segni di punteggiatura e le sigle speciali del comando "mode" che e' uno dei tanti di cui MS-DOS e' costituito.

<u>Elenco rapido dei componenti MS-DOS.</u> Per vedere di quali componenti e' fatto MS-DOS basta battere il comando **dir** [CR], avendo inserito preventivamente il dischetto sistema nel drive A (inferiore).

La risposta e' immediata e tra i vari nomi comparira' anche "mode", prima menzionato.

Per inciso, in questa rapida panoramica possiamo mettere in pratica alcune semplici e utili manovre. Ad esempio scritto cosi' il comando **dir** vi presentera' tutti i componenti di MS-DOS, ciascuno su una riga dedicata, insieme con alcune informazioni che riguardano il singolo componente. **Mode**, in particolare, e' accompagnato dall'estensione COM, la sua occupazione in byte (2382), la data (29.6.1984) e l'ora (9.00) in cui e' stato registrato sul dischetto sistema.

1.3.3 I LINGUAGGI

Abbiamo finora cercato di mettere in evidenza i legami tra i diversi strati di software esistenti in M24 e nello stesso tempo abbiamo avvicinato il concetto di sistema operativo e l'ambiente che gli equivale, e quello di risorse di un sistema e di programma applicativo.

Proponiamoci ora di dare qualche idea sui linguaggi, con i dovuti accorgimenti per mantenerci a contatto con i problemi pratici della realta' quotidiana.

Come abbiamo gia' osservato, lo svolgimento da parte dell'elaboratore di un qualsiasi compito, fosse anche del calcolo piu' semplice, viene ottenuto per mezzo di operazioni elementari in cui quel compito puo' essere suddiviso. Ognuna di queste operazioni viene trasmessa alla macchina attraverso un linguaggio sufficientemente espressivo e particolarmente privo di ambiguita'.

Come e' noto, non possiamo impiegare il linguaggio naturale perche' questo non rispetta quasi mai i criteri sopra indicati.

La ragione di cio' sta nel fatto che in una frase parlata la struttura astratta che ne ispira i contenuti viene troppo spesso contaminata da errori formali e da inutili ripetizioni: adottare quindi un linguaggio naturale per comunicare con una macchina e', allo stadio attuale dei linguaggi naturali e nel migliore dei casi, una perdita secca di efficenza elaborativa. In generale, dover tener conto di tutte le sfumature e le ridondanze provocherebbe tante e tali complicazioni da mettere subito in crisi l'elaboratore piu' dotato.

Su questo tema si potrebbero citare molti esempi: tra questi la traduzione automatica, una specie di "moto perpetuo" in campo informatico, che ha illuso non pochi celebri personaggi degli anni sessanta.

Tornando al nostro obiettivo di impartire ordini all'elaboratore e di cercare quindi un mezzo di comunicazione, constatiamo che, in fondo, un linguaggio e' come un gioco che ha le sue regole. In un gioco di societa', pero', gli avversari sono si' diversi per abilita', ma appartengono pur sempre al genere umano. Il problema, quindi, di definire un linguaggio di programmazione solleva complessita' inaudite. Nella costruzione di tale linguaggio l'aspetto comunicazione deve tener conto anche dell'enorme differenza di competenze linguistiche tra l'elaboratore e l'uomo. S'intende che dell'incompetenza linguistica della macchina non si vuol far carico alla macchina stessa, ma all'uomo che non sa dargliela. E allora, come il livello della

conversazione tra persone deve essere commisurato all'abilita' a comprendere del meno dotato tra gli interlocutori, altrettanto dovra' accadere tra uomo e macchina. Ma come far arrivare i nostri comandi allo strato piu' interno dell'elaboratore, quello strato ibrido, intriso di logica binaria, di elettronica al silicio e di fili? Ricordiamo allora le fasi logiche attraverso le quali passa l'idea guida da sviluppare prima di trasformarsi in un programma applicativo e diventare trasparente e accettabile dal linguaggio base della macchina. Esse sono, in sintesi, cosi' articolate:

1. **Analisi**, ovvero formalizzazione del problema. Ricerca della procedura logica. Individuazione e verifica dimensionale delle risorse di macchina necessarie.

2. **Progettazione** della struttura logica del programma.

3. **Stesura** del programma con la sequenza delle istruzioni secondo l'ordine logico di esecuzione. Tali istruzioni sono scritte dall'uomo in linguaggio simbolico. Correzione degli eventuali errori logici e formali.

4. **Traduzione** del programma simbolico (di cui alla fase 3) in linguaggio macchina.

Arrivati alla fase 4, ormai, il programma puo' essere compreso dalla macchina e se tutte le condizioni di incertezza e di ambiguita' sono state rimosse, puo' quindi considerarsi pronto per l'uso e l'elaborazione dei dati in ingresso secondo la procedura logica studiata nella fase 1. Rimane ancora un fatto abbastanza importante che influenza la prestazione della macchina e del quale vi parleremo rapidamente, nella speranza di non complicare troppo il discorso: quando al punto 4 abbiamo introdotto la fase di traduzione di un programma in linguaggio macchina, non abbiamo implicitamente fatto riferimento a un particolare modo in cui tale attivita' verra' effettuata.

In effetti vi sono almeno due grandi categorie di programmi di traduzione a ciascuna delle quali corrisponde un diverso modo di far lavorare la macchina:

- modo interprete

- modo compilatore.

Entrambi hanno per obbiettivo la traduzione di un programma scritto in linguaggio simbolico (detto programma **sorgente** perche e' quello scritto dall'uomo e piu' vicino alla sua mentalita') in un altro programma (detto **oggetto**) a livello piu' basso che rispetta le regole del linguaggio della macchina.

La differenza tra i due modi di tradurre il sorgente sta nella modalita' di far seguire subito oppure di rinviare in altro momento l'esecuzione del programma oggetto. Un programma eseguito in traduzione simultanea si dice che viene **interpretato**, mentre quello che viene eseguito solo quando sia stato completamente tradotto si dice **compilato**.

Poiche' un'istruzione tradotta una volta per tutte e registrata in linguaggio base nel programma oggetto non risente tutte le volte che deve essere eseguita del tempo di traduzione e' evidente che un programma gia' tradotto e' molto piu' veloce di uno interpretato, istruzione per istruzione, durante la sua esecuzione.

Tra i linguaggi evoluti, e nello stesso tempo piu' indicati per il principiante, il Basic (Beginner's All purpose Symbolic Instruction Code) e' uno dei piu' diffusi.

I programmi che troverete nei prossimi capitoli sono stati scritti in Basic per una comune configurazione M24 in ambiente MS-DOS. Nello stesso ambiente avremmo potuto scriverli in Pascal, oppure, scriverli in Pascal ma in ambiente diverso.

Abbiamo optato per il Basic non solo perche' e' il piu' diffuso, particolarmente per chi ha cominciato a programmare su piccoli home computer, ma soprattutto perche'vengono eseguiti in modo interprete.

Cio' significa, come prima dicevamo, una perdita in efficienza, ma anche la possibilita' di disporre di programmi interattivi, proprieta' fondamentale per chi sta imparando a usare un elaboratore. In tal modo ogni modifica apportata puo' essere immediatamente interpretata e valutata nei suoi effetti. Diversamente, ad ogni modifica avremmo dovuto compilare l'intero programma, o una parte di questo, e l'effetto, privo di interattivita', sarebbe stato assai meno efficace.

1.4 A COSA PUO' SERVIRE INIZIALMENTE M24

Generalmente per conoscere un nuovo strumento e' necessario aver assimilato una buona dose di nozioni fondamentali che lo riguardano e aver fatto anche un po' di esperienza. M24, come strumento informatico, non si sottrae a questa regola.

Pertanto, il metodo adottato in questo libro alterna la descrizione della macchina con esercizi di assestamento per esplorare gradualmente il campo di azione sperimentale e nello stesso tempo valutare il grado di apprendimento.

Cominciamo allora nei prossimi paragrafi con alcuni esempi di riscaldamento per vedere se riusciamo a fare su M24 quello che eventualmente sappiamo gia' fare sulle calcolatrici tascabili e sulle macchine per scrivere. Con questo naturalmente non abbiamo alcuna intenzione di convincervi a non servirvi piu' di questi utilissimi strumenti. L'idea e', quindi, quella di farvi decollare su M24 con le vostre attuali abilita', prima di lanciarvi nell'impresa di imparare a programmare.

1.4.1 COME UNA CALCOLATRICE TASCABILE

C'e' chi dice che non bisogna chiamare calcolatore un elaboratore elettronico perche' questo sa far ben altro che calcoli. Concordiamo, ovviamente, con l'obiezione, ma tuttavia, se si tratta di un microelaboratore, non possiamo non saperlo usare anche come semplice calcolatore tascabile.

Anzi, per quanto in parte sia vero, dobbiamo sfatare la leggenda che su elaboratore sia proprio impossibile fare le cose piu' semplici, come appunto eseguire dei calcoli immediati.

Per dimostrare quanto sopra vedremo proprio qualche esempio di calcolo immediato. Come su altri elaboratori, anche su M24, non e' possibile far dei calcoli immediatamente dopo averlo acceso: il motivo sta nel fatto che ci troviamo di fronte a uno strumento di razza superiore, che e' stato progettato per rispondere a domande di portata piu' ampia. Dobbiamo quindi prepararlo alla banalita' della nostra attuale richiesta.

Per dirla in altre parole, l'elaboratore non puo' fare a meno di passare attraverso le fasi di apertura abituale, o come dicono gli addetti, di inizializzazione. Occorre cioe' fargli compiere le operazioni preliminari gia' viste per portarlo fino allo strato di operativita' Basic.

Innanzitutto, prima di accendere M24 facciamo questi semplici controlli:

1. Il cavo del video deve essere collegato all'unita' base

2. La spina (con contatto di terra laterale) all'estremita' del cavo di alimentazione di M24 deve essere inserita in una presa compatibile di energia elettrica (controllare anche che la tensione sia giusta).

3. Se lavorate con la stampante, la spina del suo cavo di alimentazione deve essere collegata a una presa di corrente. Occorre allora un multiplatore di presa. Inoltre le spine alle estremita' del cavo di comunicazione tra base M24 e stampante devono essere correttamente fissate nei rispettivi bocchettoni.

4. Un dischetto sistema deve essere inserito nel drive inferiore di M24, quello designato convenzionalmente con la lettera **A.**

Terminati questi controlli possiamo finalmente accendere M24 e toccarne i "bottoni" osservandone le fasi di risveglio che dovranno avvicendarsi nel seguente modo:

- si illuminano alternativamente le lampadine rosse della tastiera;

- si illumina la lampadina rossa situata vicino alla fessura del drive in cui avete inserito il dischetto sistema;

- dopo le fasi di autodiagnosi e di caricamento del nucleo dell'MS-DOS appaiono le "videate" di figura 1.10 nelle quali il sistema vi chiedera' prima di introdurre la data e poi l'ora; se non volete cambiarle e' sufficiente ignorare le domande e battere due volte il tasto [CR]. Finalmente le fasi preparatorie hanno termine e appare il noto segnale ">" di riconoscimento dell'MS-DOS, preceduto dal numero del drive dal quale e' stato estratto il nucleo.

A questo punto il sistema operativo MS-DOS e' ai vostri ordini e potete chiedergli di fornirvi uno dei suoi moduli piu' pregiati: l'interprete del linguaggio Basic.

Battete pertanto la parola [**gwbasic**] senza dimenticarvi che, dopo aver impostato questa sigla sulla tastiera, bisogna anche convalidarla, spedendola all'unita' centrale.

Per far cio' battete il solito [CR], il tasto grande a L rovesciata. Vedrete immediatamente accendersi la lampadina rossa del drive A (imparate a seguire i messaggi elementari che provengono da queste lampadine spesso trascurate). Cio' significa che la parte di MS-DOS residente in memoria centrale ha decodificato il vostro comando [gwbasic], lo sta eseguendo con la lettura sul dischetto in A dei circa 70000 byte dell'interprete Basic, e contemporaneamente lo sta trasferendo in memoria centrale.

Alla fine del trasferimento, l'interprete Basic fara' atto di presenza con il suo segnale di riconoscimento [Ok]. Non dimenticate l'utilita' di questo piccolo ma importante dettaglio di identificazione: cio' che potete chiedere a M24 in ambiente Basic e' diverso da cio' che potete ottenere in MS-DOS. Orbene, Basic sta aspettando di soddisfare le vostre richieste. Sapendolo fare, potreste gia' scrivere un programma: ma di questo tratteremo gradualmente nei prossimi paragrafi.

Vediamo invece come si fa di conto su M24. Innanzitutto occorre informare M24 che vogliamo eseguire dei calcoli:per far cio' bisogna introdurre il simbolo [?].

Questa operazione viene fatta con estrema facilita' mantenendo premuto il dito medio della mano destra sul tasto SHIFT (quello alla destra del tasto lungo per lo SPAZIO) e agendo con l'indice della stessa mano sul tasto [?], che e' immediatamente alla sinistra del tasto SHIFT prima menzionato.

Controllate che sul video sia veramente apparso il carattere [?]. Se qualcosa non e' andata nel verso giusto, per esempio avete rilasciato lo SHIFT prima del tasto [?], allora avrete il problema di cancellare il carattere [/] che sara' comparso al posto del [?]. Occorrera' allora cancellare i caratteri errati col tasto [←], oppure col tasto DEL (in basso a destra) che vi spazzera' via il carattere corrispondente alla posizione che occupa il cursore. In qualsiasi modo siate riusciti a procurarvi il simbolo [?] per farlo apparire in modo "pulito" sul video, ora dovrete avere un calcolo in mente da fare eseguire a M24.

Gli operatori aritmetici [+ - * /] si trovano insieme con le cifre nella tastiera numerica. Il simbolo [*] e' convenzionalmente adottato in Basic (e in altri linguaggi) come operatore per la moltiplicazione.

Battiamo, finalmente, gli operandi e l'operatore prescelto per il tipo di calcolo che volete eseguire. Per esempio: 152 diviso 326 che verra' battuto 152/326.

L'impostazione viene, come sempre, seguita a vista sullo schermo: qualunque errore di battuta puo' venir corretto come prima indicato col tasto [←] o col tasto DEL. Quando tutto combacia, sul video dovra' comparire:

```
Ok <-------- segnale di presenza Basic

     +--------operatore di divisione
?152/236
↑ ↑   ↑
| └───┴──── operandi
|
└simbolo per chiedere al Basic di fare calcoli immediati.
```

Arrivati a questo punto concludete l'operazione battendo il tasto speciale [CR] e otterrete il risultato nella riga immediatamente sotto quella in cui avevate posto gli operandi. Dopo aver scritto il risultato [.6440678], il Basic si ripresenta con il suo Ok seguito dal solito cursore lampeggiante.

Viene ora spontaneo chiedersi il motivo di tante operazioni di attrezzaggio per arrivare a fare cio' che una comune calcolatrice tascabile fornisce subito senza eccessive predisposizioni.

La risposta e' ovvia: abbiamo semplicemente a che fare con un elaboratore che usa col mondo esterno un certo linguaggio di comunicazione le cui regole grammaticali sono di gran lunga piu' articolate ed espressive di quelle che possiamo trovare in una calcolatrice. Questo insieme di regole esiste anche nei linguaggi piu' semplici come il Basic: pertanto, se vogliamo trasmettere qualcosa all'elaboratore dobbiamo superare il livello di difficolta' costituito dalla struttura formale del linguaggio di comunicazione.

Torniamo ora a far pratica di calcoli immediati. Se si vuole , ad esempio, calcolare "2 elevato alla 15", si deve scrivere:

```
? 2^15 [CR]
```

Si noti che la notazione esponenziale viene espressa con l'operatore [ ]. Questo simbolo e' riprodotto sul cappuccio del tasto [6] e viene quindi generato premendo il tasto speciale [↑] o SHIFT. Il risultato (32768) appare sul video nella riga sottostante immediatamente dopo la pressione del tasto [CR].

Provate ora a inserire una sequenza piu' articolata di operandi e di operatori aritmetici, ovvero, un'espressione. Per esempio:

```
? 25*15/3+2
```

Cosi com'e' scritta anche voi avreste delle incertezze nell'ordine di precedenza da adottare nell'eseguire i calcoli, a meno che non abbiate le vostre convenzioni. In sintesi i modi per interpretare quell 'espressione sono almeno quattro. Vediamo i passi di calcolo in ciascuno di essi:

```
1)  25*15=375
    375/3=125
    125+2=127

2)  15/3=5
    25*5=125
    125+2=127
```

3) 3+2=5
   15/5=3
   25*3=75

4) 3+2=5
   25*15=375
   375/5=75

Avrete notato che dei quattro modi di interpretare l'espressione due conducono a un risultato diverso: non e' quindi solo questione di di forma, ma anche di sostanza. Dovremo, pertanto conoscere quali convenzioni adotta il Basic per assegnare le precedenze ai diversi operatori. Proviamo a impostare il calcolo battendo di seguito

```
? 25*15/3+2 [CR]
```

Il risultato e' 127: cio' significa che il prodotto e la divisione hanno precedenza sull'addizione. In sintesi, per gli operatori numerici la regola che il basic M24 segue per stabilire la precedenza e' la seguente:

1. Elevazione a potenza (operatore:^)

2. Moltiplicazione e divisione (operatori: * /)

3. Addizione e sottrazione (operatori: + -).

Per operatori con lo stesso ordine di precedenza o priorita', vale inoltre la posizione che occupano nell'espressione: infatti il Basic assegna la precedenza all'operatore che incontra prima, nell'ordine in cui legge l'espressione da sinistra a destra. Le precedenze possono essere infine modificate a piacere con l'impiego opportuno delle parentesi, come in matematica. Riassumendo, tenendo conto delle precedenze tra operatori, delle parentesi e del senso di lettura, possiamo concludere che l'ordine di esecuzione interno del Basic nel fare i calcoli e' il seguente:

- esamina l'espressione da sinistra a destra
- calcola le espressioni racchiuse dentro le parentesi
- esegue le elevazioni a potenza
- esegue le moltiplicazioni e le divisioni
- esegue le addizioni e sottrazioni.

Nello scrivere delle espressioni in Basic si deve naturalmente ricordare di sostituire alle abituali regole matematiche di scrittura quelle proprie del linguaggio. Ad esempio, l'espressione

$$\frac{4x + 5(x-3y)}{x - y}$$

in Basic dovra' essere scritta cosi':

(4*x+5*(x-3*y))/(x-y).

Per dar maggior chiarezza alle espressioni, le parentesi possono essere usate anche se non strettamente necessarie. Si ricordi inoltre che se un'espressione contiene una variabile alla quale non e' stato dato preventivamente alcun valore, nel calcolo dell'espressione, il Basic le assegna il valore zero.

Facciamo ora un esempio di applicazione per comprendere bene le regole di priorita' che abbiamo prima esaminato. Per ciascuna delle espressioni riportate qui di seguito viene dettagliato l'ordine con il quale il Basic prende in esame i vari termini e la priorita' di esecuzione per le varie operazioni.

```
Esempio 1     ? 2+2^5/2*8*(3^(3+1))          = 10370

Esempio 2     ? 2+2^5/2*8*(3^3+1)            =  3586
```

Nel primo esempio l'operazione con la massima precedenza e' quella contenuta dentro le parentesi piu' interne. Si comincia quindi a fare 3+1=4. Successivamente, ripetendo daccapo la scansione dell'espressione -sempre da sinistra verso destra- andando alla ricerca di altre parentesi, troviamo quella che contiene gia' il risultato della prima operazione, cioe' 3 ^ 4=81. Nelle ulteriori scansioni gli operatori interessati, dopo quello di elevazione a potenza (2 ^ 5), sono quelli delle moltiplicazioni e della divisione. Eseguiremo pertanto 2 ^ 5=32 e l'espressione si sara' a questo punto trasformata nella seguente:

```
2+32/2*8*81
```

e, ancora:

```
2+16*8*81
2+128*81
2+10368
10370
```

Come avrete notato, l'ultima operazione presa in esame e' quella dell'addizione.

Nel secondo esempio invece un'operazione di addizione e' al secondo posto, dopo l'elevazione a potenza, perche' compare in un'espressione racchiusa tra parentesi che, come abbiamo prima detto, vengono esaminate per prime. Allora la sequenza di esecuzione per questo secondo esempio sara':

```
2+2^5/2*8*(27+1)
2+2^5/2*8*(28)
2+32/2*8*28
2+16*8*28
2+128*28
2+3584
3586
```

Un paio di parentesi hanno ridotto a meno di un terzo il risultato!

L'uso corretto degli operatori e delle notazioni e' quindi della massima importanza.

Un altro aspetto di rilievo nel trattare espressioni numeriche e' conoscere bene la tecnica degli arrotondamenti che la macchina necessariamente applica sul risultato di certe operazioni. Questa tecnica non e' un arbitrio della macchina, ma una precisa necessita', anzi, abusando delle parole, "una necessita' di imprecisione".

Spiegamoci meglio. A nessuno sara' sfuggito nel fare le moltiplicazioni la proprieta' del prodotto di avere un numero di cifre della parte intera maggiore di quella delle parti intere di ciascuno dei fattori: anzi, la regola e' che il numero delle cifre della parte intera di un prodotto e' la somma di quelle delle parti intere dei fattori. "Sempre ?" potreste chiedere. Non sempre. Si puo' dimostrare che non supera mai la somma delle cifre delle parti intere dei fattori, salvo nei casi in cui, per l'esiguita' del numero espresso da uno o da entrambi i fattori, il prodotto ha un numero di cifre della parte intera pari a quella somma meno uno. Un esempio su numeri facili (ma la regola e' generale):

2 x 5 = 10   ma   2 x 3 = 6.

Premessa questa importante osservazione sulla "lievitazione" delle cifre, si puo' capire che per calcoli con operandi molto "grandi" anche M24 sia in difficolta' per stampare il risultato. Infatti, **normalmente**, per risparmiare memoria si utilizza la versione che prevede come massimo la possibilita' di stampare un numero di 7 cifre (notazione in semplice precisione).

E per i numeri che superano le 7 cifre? M24, come molti altri elaboratori, utilizza la **notazione scientifica** o esponenziale, che e' un modo di scrivere i numeri utilizzando le potenze in base 10.

Esempio: per il risultato della moltiplicazione 512x32768=16777216 (numero di 8 cifre) M24 scrivera' 1.677722E+07 ove E+07 significa 10 elevato a +7. Cioe' il numero delle cifre significative (mantissa) deve essere in questo caso moltiplicato per dieci milioni (10 ^ 7= 10000000).

Come si potra' notare 16777220 (=1.677722x10000000) e' arrotondato rispetto a 16777216 ma ha conservato l'ordine di grandezza. (Ecco il perche' di "semplice precisione"). Volendo una notazione in doppia precisione con mantissa a 16 cifre, M24 e' in grado di fornircela utilizzando pero' maggior memoria.

Sul modo di trattare i numeri torneremo in seguito nei prossimi capitoli, dove tratteremo anche la possibilita', prima accennata, di cambiare dentro M24 la precisione con cui vogliamo ottenere i risultati.

Ricordiamo infine che i problemi sopra accennati sono alla base dell'elaborazione elettronica: ignorarli o comprenderli solo in parte e' di grave pregiudizio per ogni uso corretto della macchina. Tra l'altro, l'argomento e' materia di insegnamento nelle scuole superiori e viene chiamato "Calcolo numerico": ne rimandiamo pertanto l'approfondimento ai relativi testi specializzati.

CAPITOLO
DUE

IMPARARE A PROGRAMMARE

Quando l'uomo comincio' a razionalizzare il progetto di un elaboratore elettronico non prevedeva certo che in cosi' pochi anni il processo tecnologico avrebbe tanto progredito; ne' era possibile stimare quanto questo progresso avrebbe stimolato una maggiore ricerca nel campo del software. Sta di fatto che buona parte del guadagno realizzato dall'elevato tasso di sviluppo della tecnologia del silicio viene continuamente smorzato dagli elevati costi per la progettazione di nuovi sistemi operativi e per la loro manutenzione.

Si stima che dal 1955 ad oggi il rapporto prezzo/prestazione sia migliorato di oltre 200 volte (ogni due o tre anni la prestazione si raddoppia a parita' di costo).

Non altrettanto puo' dirsi per le tecniche di produzione del software che al massimo sono migliorate del 3% all'anno; il che significa che in quasi trent'anni il rapporto sopra indicato ha guadagnato solo 2 volte e mezzo. Evidentemente cio' dipende dal fatto che i programmi devono essere ancora scritti a mano; e lo saranno finche' non sara' messo a punto un "ambiente" in cui i calcolatori quasi si scriveranno i programmi da soli, in virtu' di linguaggi cosi' evoluti che con poche istruzioni comunicheranno tutto all'elaboratore.

In effetti, a tutt'oggi, i modesti miglioramenti ottenuti nella scrittura dei programmi, anche con l'uso di potenti compilatori, sono largamente attutiti dall'elevato costo della mano d'opera. Inoltre, nello sviluppo del software, specialmente di quello applicativo, intervengono troppi strati professionali tra l'idea e il prodotto finito. Sarebbe gia' un grosso risultato se la prima formalizzazione dell'algoritmo in cui l'idea viene trasmessa, fosse compiuta da chi conosce bene la materia.

Cio' premesso, accostiamoci all'arte della programmazione.

SCOPI DEL CAPITOLO

Gli obiettivi di questo capitolo sono molteplici.

1. Fornire le notizie base indispensabili per la comprensione del processo di generazione di un programma.

2. Comprendere alcuni concetti fondamentali della programmazione come la preparazione dell'ambiente operativo e delle sue tipiche fasi di lavorazione.

3. Conoscere quali sono le risorse di cui dispone l'ambiente di programmazione.

4. Sviluppare una conoscenza pratica, anche se non completa per ora, del linguaggio GWBASIC.

5. Assimilare alcune principali regole di punteggiatura e i principali comandi del linguaggio.

2.1 COME NASCE UN PROGRAMMA

Quando si pensa alle calcolatrici programmabili si corre per analogia ad altri strumenti programmati. Tra questi uno dei piu' diffusi e' il telecomando del televisore. Non tutti sanno che l'associazione tra canale e tasto non e' rigida. Ad esempio, se sono 20 i tasti dedicati ad altrettanti canali, non e' il costruttore che li sceglie e li "salda" al tasto. Infatti, quando lo desideriamo, possiamo con opportune manovre riassegnare le 20 predisposizioni abituali tasto-canale. Cio' e' possibile perche' in quel minuscolo apparecchio, oltre alle capacita' di "sparare" un fascio di infrarossi contenente il messaggio per il televisore (che in qualche modo dopo averlo decifrato lo rende operativo per il cambio del canale), c'e' anche un sorprendente aggregato di elettronica che esegue istruzioni per cambiare le 20 accoppiate tasto canale.

Anche senza leggere le istruzioni del fabbricante, o essere informatici, si intuisce che tale telecomando deve essere dotato di memoria e di qualche organo di calcolo (un microprocessore). Tale memoria ricorda tutti i canali precedentemente selezionati corrispondenti ai tasti di cui il telecomando e' dotato.

Molti telecomandi per risparmiare hanno un tasto di preselezione (shift) a due posizioni: in tal modo si raddoppiano i canali selezionabili, perche' ogni tasto puo' chiamare due stazioni.

Altri esempi di programmi casalinghi potreste trovarli negli elettrodomestici, nella sveglia dell'orologio digitale, etc.

Ma torniamo agli elaboratori. Vediamo ora come viene concepito un programma applicativo.

Generalmente, un programma nasce con un'idea-guida. Lo sviluppo successivo e la definizione della sua struttura devono essere continuamente verificati con gli obiettivi del problema. Inizialmente non c'e' spazio per i dettagli, almeno per quelli che intuitivamente non modificano la sostanza. In questa fase dobbiamo anche severamente analizzare i vantaggi che possono derivare dalla meccanizzazione della procedura, rispetto alla fatica di realizzazione del programma. Quando questa analisi di fattibilita' ci convince, si passa al vero e proprio piano di progettazione: formuliamo ipotesi di struttura e nello stesso tempo cominciamo a tener conto dei dettagli e delle risorse di macchina necessarie.

In particolare, occorre stabilire le dimensioni della memoria, centrale e ausiliaria, idonea a ospitare il programma ed eventualmente i dati in ingresso e in uscita.

Un aspetto essenziale in questa fase consiste nel saper valutare validi, o al massimo adattabili al problema in esame, gli schemi risolutivi che conosciamo. Spesso, invece, non riutilizziamo schemi gia' collaudati per ricercarne di nuovi. In questo modo di procedere

-che non dovremmo facilmente accettare- risiede una delle maggiori cause di ritardo nella realizzazione di un programma.

D'altra parte, nella reciproca influenza tra scelte progettuali e materia da meccanizzare sta la base di un buon programma. Inoltre, c'e' un motivo che attrae a programmare chi non sia di estrazione informatica: la prospettiva di imparare quanto basta per liberarsi da situazioni culturalmente subalterne; in tal modo si puo' riprendere il contatto completo con la sostanza della propria materia alla quale in parte si rinunciava per la difficolta' a trasmetterla agli informatici puri.

Orbene, i programmi sono fatti di istruzioni che qualcuno ha scritto per la macchina, trasferendo punto per punto tutti i particolari di un certo processo mentale, suddiviso in parti o procedure analizzate accuratamente da chi conosce il problema nei suoi piu' sottili contenuti.

Per inciso, la domanda che qui viene spontanea e' se l'ideatore (analista) della procedura formalizzata (algoritmo) debba coincidere con lo scrittore del programma (programmatore) o se possano essere persone diverse.

In pratica, chi conosce bene la materia puo' non sentire molto la vocazione di scrivere programmi. Viceversa il programmatore non sempre ha la conoscenza di base della materia che si desidera meccanizzare. Ovviamente e' auspicabile un minimo di interdisciplinarieta' tra le professioni di analista e di programmatore.

Comunque sia, supponiamo di avere un programma in memoria e tentiamo di fare una prima ispezione superficiale sulla sua struttura e sui movimenti interni nei confronti della macchina che lo ospita.

Un programma e' costituito da una successione di **istruzioni** che la macchina legge in un certo ordine, interpreta, modifica ed esegue. Si noti che la memoria centrale non contiene solo il programma, ma anche i dati che questo deve elaborare.

Chi da' vita a questo processo? L'uomo, con pochi ma essenziali **comandi** esterni alla macchina.

Facciamo osservare che abbiamo usato la parola comando e non istruzione perche' assai diversi sono i criteri operativi che queste due parole racchiudono. Vedremo in seguito che la differenza sostanziale sta nel fatto che agiscono in tempi diversi, pur essendo entrambi ordini per la macchina: l'istruzione e' un comando programmato (sta dentro un programma) che verra' eseguito a tempo debito, quando l'intero programma verra' eseguito.

Un altro concetto importante da chiarire e' la possibilita' che un programma possiede di modificarsi nel suo interno, non solo in rapporto ai dati introdotti -il che in parte e' ovvio- ma anche in relazione ai risultati di una certa elaborazione intermedia.

In questo fatto risiede la maggiore differenza sul piano logico tra il programma di un elaboratore e il rullo di una pianola. In quest'ultima, infatti, non sono ammessi cicli interni, se non attraverso l'effettiva ripetizione sul rullo del motivo musicale. Ne' e' possibile far fare al rullo dei salti in avanti o indietro, perche' lo svolgimento del rullo puo' avvenire solo in un senso e, normalmente, non ne e' previsto il riavvolgimento o l'avanzamento rapido.

Ora bisogna fare un altro sostanziale rilievo: un elaboratore -l'abbiamo visto- puo' comportarsi come una calcolatrice tascabile,

ma e' nato fondamentalmente per altri scopi.

Un calcolo comporta sempre la necessita' di introdurre i dati di partenza su cui la macchina deve operare. Se il calcolo deve essere eseguito poche volte non conviene programmarlo su un calcolatore, a meno che la procedura non sia di eccezionale complessita', che' vi sarebbe troppa differenza tra il tempo di introduzione degli operandi e il tempo di esecuzione. Se invece quel calcolo, all'interno di un programma, deve essere ripetuto n volte, dove n puo' essere un numero anche molto elevato, in tal caso l'impiego di un elaboratore si giustifica perche' il tempo di esecuzione complessivo raggiunge l'ordine di grandezza del tempo di introduzione dei dati di partenza.

A questo punto, avrete certamente compreso che l'idea-guida di un programma deve essere valutata criticamente anche sul piano dell'economicita' oltre che essere sviluppata con un certo metodo. Per metodo non intendiamo solo un atteggiamento puramente razionale rivolto principalmente all'efficienza e all'organizzazione del lavoro: la mia personale convinzione e' che quando si progetta un programma conviene sempre alternarsi tra due atteggiamenti operativi complementari del pensiero, una sorta di miscela di fantasia e razionalita'. Sulle scelte cosi' operate cominciano a prender forma gli effetti desiderati che la macchina dovra' produrre quando eseguira' il programma.

Il modo per provocare questi effetti sta nel saper combinare tra loro le istruzioni di un programma con le necessarie risorse di macchina e tutto cio' va comunicato in un opportuno linguaggio. Una prima traccia di massima da seguire per la progettazione di un programma e' la seguente:

1. Quali dati di partenza servono? (Dati e funzioni operative di macchina).

2. Da dove provengono questi dati? Da tastiera. Da sistema operativo. Da dischetto: a volte da piu' di un dischetto, con conseguenti operazioni di alimentazione dischetti (sfilare un dischetto e metterne un altro).

3. **Come** elaborare quei dati?

4. Dove devono uscire i risultati? Su video, su carta, su dischetto, verso il sistema operativo.

5. Con quale stile far uscire i risultati? Criteri di impaginazione per i testi. Progetti grafici per gli effetti statici e dinamici da associare al risultato.

Ovviamente il "come " del punto 3 richiede di avere sempre in mente l'obiettivo o i prodotti intermedi che soddisfano l'idea guida. Gli aspetti relativi allo stile non sono un punto trascurabile, perche' sono legati all'interpretazione o alla facile lettura dei risultati.

Anche se sarete solo voi i fruitori del programma che state progettando, un buon stile di presentazione dei risultati ve ne fara' ricordare il corretto impiego. Osservazione analoga andra' fatta in

margine al punto 2, in merito all'introduzione dei dati da tastiera mediante l'ausilio di opportune maschere guida (anche queste vanno programmate).

Per quanto riguarda la procedura elaborativa -o algoritmo- e' opportuno cominciare a individuarne le parti essenziali:

- il corpo principale,
- le fasi di attivita' in cui l'algoritmo puo' essere scomposto (calcoli interni, operazioni di lettura o di registrazione su dischetti, operazioni di acquisizione dati da tastiera, stampa di risultati, etc.),
- i dati in ingresso (**input**),
- i risultati finali desiderati (**output**),
- le alternative della procedura, ovvero le principali proposte per l'utente,
- eventuali proposte secondarie.

Di tali fasi occorre anche conoscere la successione, nell'ambito di uno studio completo del piano temporale delle attivita' meccanizzate e degli interventi dell'operatore.

Riprendiamo ora quell'esempio di falegnameria che abbiamo gia' trattato per introdurre il concetto di ambiente; adesso pero' preoccupiamoci non tanto degli strumenti necessari (ambiente) quanto del contenuto stesso della lavorazione, nella sua logica interna fatta di scelte, di attese, di verifiche etc.

In questo nuovo modo di porre il problema, quindi, ha piu' rilevanza analizzare **come** praticare un foro nel legno piuttosto che la definizione degli utensili e del loro attrezzaggio. Intendiamoci: non e' che, siccome vogliamo formalizzare la procedura, per questo il ruolo dell'ambiente sia diventato meno importante. Sono semplicemente due aspetti della stessa realta'.

Tracciamo quindi il flow chart della lavorazione seguendo la logica successione degli atti che conducono al risultato, in questo caso costituito da un foro in un certo pezzo di legno. Notate che l'idea guida e' stata appena abbozzata. Le uniche cose che sappiamo sono: un foro e un pezzo di legno.

Per togliere incertezza e ambiguita' al discorso occorre almeno definire:

- diametro del foro
- forma del pezzo
- localizzazione del foro sul pezzo
- interazione tra pezzo e ambiente

Nella figura 2.1 oltre alla forma del pezzo sono disegnati i due modi di fare il foro nel pezzo: la scelta dipende dall'eventuale interferenza tra il pezzo e il mandrino.

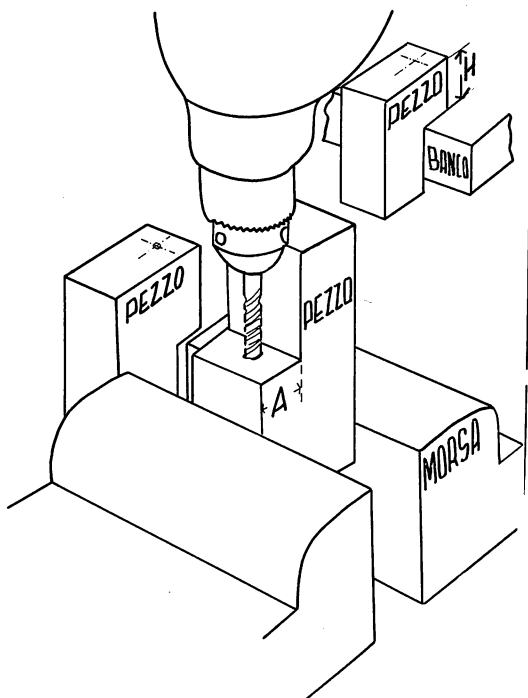



Figura 2.1

Infatti nel caso A la vicinanza del foro allo spigolo interno del pezzo non consente la foratura. Si richiede quindi di rovesciare il pezzo e forarlo dall'altra parte

E a questo punto sorge un altro dilemma. Come afferrare il pezzo durante la lavorazione? Si puo' afferrarlo in due modi: con la morsa oppure, piu' semplicemente, accostando il pezzo allo spigolo del banco. Nel caso A, quand'anche fosse praticabile, e' opportuno l'impiego della morsa.

Dal diagramma delle fasi riportato nella figura 2.2 vediamo che in ogni caso si arriva al punto D, dal quale puo' iniziare una sequenza di attrezzaggi relativi all'esatta segnatura del pezzo nel punto da forare, e al serraggio nella morsa o piu' semplicemente l'accostamento contro lo spigolo del banco.

Seguono le fasi di attrezzaggio del trapano -scelta della punta in relazione al diametro del foro e all'altezza del foro, e numero di giri- e finalmente la foratura.

Orbene, questo procedimento cosi' pedantemente analizzato costituisce lo schema della lavorazione, un modo non ambiguo di tradurre in termini precisi l'idea guida. Se avessimo potuto esprimerla anche in simboli formali avremmo costruito un algoritmo.

E' venuto il momento di dare anche qualche riferimento storico alla materia e un minimo di inquadramento e di terminologia.

ALGORITMO: termine derivato da al-Khuwarizmi, *soprannome* del matematico arabo Muhammad ibn Musa.

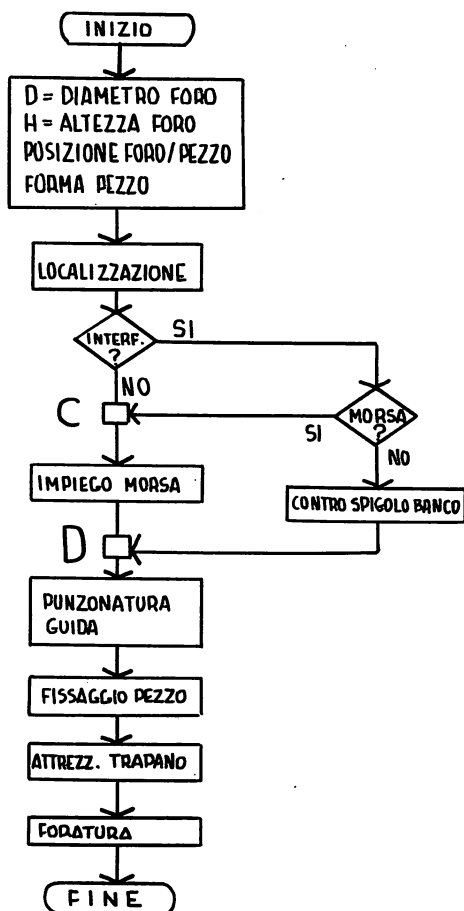



Figura 2.2

Tale termine fu usato nel Medioevo specialmente per indicare i procedimenti di calcolo numerico basati sull'uso delle cifre arabe e attualmente si usa per qualsiasi schema di calcolo. Quanda si parla, ad esempio, dell'algoritmo della divisione o della moltiplicazione, s'intende la nota disposizione delle cifre che si usa per effettuare le operazioni. Analogamente la ricerca del massimo comun divisore per mezzo di divisioni successive prende il nome di algoritmo del massimo comun divisore (detto anche algoritmo di Euclide).

DATO: nel linguaggio corrente significa quantita' di cosa data, fatto o principio che serve come punto di partenza. In informatica ha assunto un senso specifico -gli inglesi usano il plurale latino **DATA**- e viene usato per designare tutti gli elementi, numeri, lettere e simboli, o gli elementi che si riferiscono o descrivono un oggetto, idea, condizione, situazione o altri fattori.

Indica gli elementi base dell'informazione che possono essere elaborati o prodotti dalla macchina. Talvolta con il termine "dato" si intende qualche cosa che puo' essere espresso solo in forma matematica.
INFORMAZIONE: atto dell'informare, dare la forma, indirizzare, impiantare, ragguagliare, istruire. Notizia, conoscenza. Richiama espressioni come: ufficio informazioni, senza notizie, privo di informazioni, ultime notizie. In senso informatico richiama il concetto di dato in senso piu' ampio e viene pertanto riferito piu' frequentemente a raccolta di fatti o di altri dati, in particolare quella risultante dall'elaborazione di dati.

Il glossario delle voci impiegate in informatica e' da sempre un libro aperto, senza fine, tanti sono i neologismi, quasi tutti in inglese, che si aggiungono ogni giorno. Cercheremo di usarne per lo stretto necessario. Vi sono poi altri termini, presi a prestito dalla matematica, che e' bene conoscere senza giri di parole col loro vero nome, per non farci prendere alla sprovvista quando li useremo nella scrittura di un programma. Tra questi termini due in particolare devono essere bene assimilati: variabile e funzione.

Definiamo questi vocaboli fondamentali, il cui significato e' spesso distorto quando li usiamo nel linguaggio corrente. La matematica, ancora una volta, fa testo e ce ne chiede ragione appena siamo capaci di esprimerci in modo simbolico, ad esempio nelle espressioni algebriche.

ALGEBRA: Il vocabolo e' derivato dalla parola araba al-giabr coniata dal solito al-Khuwarizmi, a significare l'operazione per la quale, allorche' in uno dei due termini di un'eguaglianza compare un termine da sottrarsi, esso puo' invece venir aggiunto all'altro membro. Per esempio 5=7-2 equivale a 5+2=7 e in generale a=b-c puo' essere ugualmente espressa nella forma a+c=b.

VARIABILE: In matematica si contrappone a "costante" e si usa come aggettivo sostantivato a designare ogni quantita' suscettibile di assumere valori diversi.

FUNZIONE: Il termine e' evidentemente in rapporto con il significato che gli attribuiamo nella vita quotidiana.
Ma, in matematica, il termine ha un impiego piu' specifico e non ha tanto rapporto con oggetti separati, quanto con le loro classi di appartenenza. Il concetto di funzione e' quindi un'astrazione per definire la classe di appartenenza di certe entita'. Una definizione piu' generale si riferisce ai legami tra due insiemi di elementi: cosicche' per funzione s'intende un nuovo insieme costituito dalle coppie di elementi appartenenti agli insiemi di partenza, ma tali che uno stesso elemento del primo insieme non possa mai essere accoppiato con elementi **diversi** del secondo.
In tal modo, ad esempio, l'insieme delle coppie estratte da quelli dei fratelli e delle sorelle puo' non soddisfare la definizione appena data e quindi non costituisce una funzione, ma una semplice relazione.

Al contrario, dagli insiemi delle madri e dei figli possiamo individuare coppie tali da formare una funzione.
Il concetto di funzione e' oggi uno dei piu' generali che dominano tutte le scienze, specie quelle sperimentali, con numerosissime applicazioni.
Ad esempio, dalla legge di Boyle-Mariotte relativa ai gas perfetti e espressa dalla formula p=k/v ricaviamo che la pressione che esercita una certa quantita' di gas sulle pareti interne di un recipiente che la contiene e' tanto piu' grande quanto minore e' il volume del contenitore.

Il vocabolo funzione apparve per la prima volta in senso matematico in uno scritto di Leibniz del 1692, ma fu il genio di Descartes che per primo nel 1637 lo uso' come metodo per risolvere con l'algebra i problemi della geometria.

E dal metodo pratico all'impiego matematico facciamo un balzo nei secoli fino ai nostri giorni, quando per scopi informatici e' diventato quasi un termine di uso domestico. Come abbiamo gia' osservato, un elaboratore serve soprattutto a ripetere uno stesso schema di calcolo o algoritmo. L'algoritmo per il calcolo della paga di un operaio non cambia passando dalle ore lavorate dall'operaio Bianchi a quelle dell'operaio Rossi, se fanno lo stesso tipo di lavoro...e se soprattutto hanno totalizzato in un dato periodo di tempo lo stesso numero di ore lavorate. Esiste quindi una relazione tra paga e ore lavorate. Esprimiamole con dei nomi simbolici piu' brevi, per comodita': P per paga in lire e L per quantita' di lavoro espressa in ore.
Sappiamo ora tutti comprendere che al crescere delle ore lavorate aumenta la retribuzione, ma non tutti accetterebbero di esprimere questa relazione verbale in una di tipo piu' formale e meno ridondante come: P= k x L.

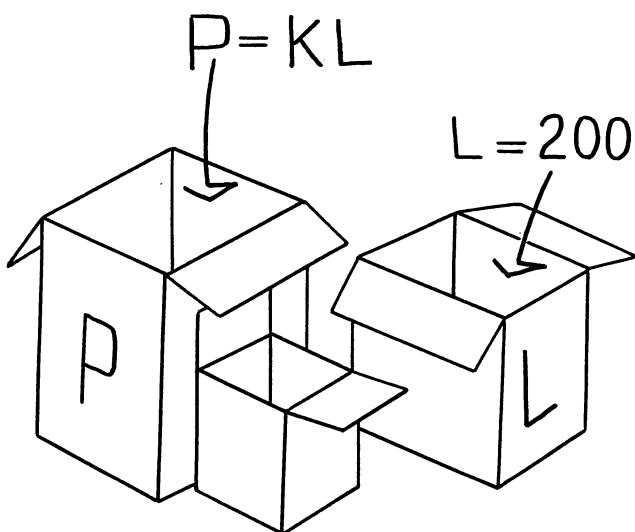



Figura 2.3

In questa relazione possiamo sintetizzare molti casi di retribuzione: a k assegniamo valori che esprimono la categoria del lavoratore, qualcosa che e' indipendente dalla quantita' delle ore prestate ed e' piu' legata al tipo e alla qualita' del lavoro svolto in base all'esperienza e all'anzianita' della persona. Ad esempio se assumiamo k=4000, la relazione diventa P=4000 L e servira' a calcolare la paga di una certa categoria in **funzione** delle ore lavorate. Per un'altra categoria k sara' diverso, e cosi' via. Si dice che P dipende funzionalmente da k a parita' di L oppure da L a parita' di k.
P e' allora una variabile dipendente, mentre k e L sono variabili indipendenti, nel senso che dipendono dai valori che di volta in volta l'utilizzatore desidera assegnar loro nella formula.

Per assestare piu' stabilmente questi concetti e farli diventare piu' familiari, diamo loro una rappresentazione meno teorica e piu' pratica e **nello stesso tempo** informatica.

Orbene, cominciamo a pensare alle variabili come se fossero delle scatole, dei contenitori, delle parti di memoria di M24. Come illustrato in fig 2.3, se assegnamo il nome L a un'area di memoria della macchina, possiamo riferirci a tale nome come al simbolo di una variabile alla quale attribuire di volta in volta valori diversi. Le modalita' per dare un valore ad una variabile in un'espressione algebrica sono di semplice sostituzione. E in un elaboratore? La risposta dipende da macchina a macchina e da linguaggio a linguaggio.. Ad esempio, in un programma Basic M24, l'istruzione Input L pone la macchina in attesa che l'operatore decida quale valore numerico attribuire alla variabile L, ovvero quale numero introdurre in quell'area di memoria di nome L.

Si noti che il nome L viene **automaticamente** assegnato dalla macchina a una sua parte di memoria, senza che l'utilizzatore debba minimamente preoccuparsi di **dove** sia fisicamente. La creazione di una variabile avviene semplicemente inventando un nome ed impiegandolo in opportune istruzioni. Alcune di queste usano particolari convenzioni per modificare il valore che le variabili hanno in un certo punto del programma. Ad esempio L=200 e' proprio la notazione che useremo in Basic per trasferire il valore 200 in quell'area di memoria etichettata col nome L. Questa istruzione viene detta di assegnazione perche' **nel momento in cui verra' eseguita** l'interprete del Basic le assegnera' il valore 200.

Conviene fin d'ora prender nota che ogni istruzione di assegnazione distrugge il precedente valore della variabile. Per evitare tale perdita possiamo trasferire il contenuto della variabile-scatola in un'altra scatola. E' sufficiente scrivere di seguito:

```
L=200 : L1=L : L=300
```

Notiamo che in linguaggio Basic si tratta di tre istruzioni di assegnazione tra loro divise dal separatore [:].

La prima istruzione assegna ad L il valore 200; la seconda travasa il contenuto di L in L1 (notare che in tal modo abbiamo automaticamente creato una nuova variabile dal nome L1). Da quel momento il valore di L1 e' 200 e abbiamo anche salvato il contenuto di L, a cui ora potremo attribuire il nuovo valore 300 con la terza istruzione L=300. Non abbiamo, ovviamente, esaurito l'argomento

delle variabili e delle funzioni che riaffronteremo in seguito.

Riprendiamo ora la metodologia di sviluppo dell'idea guida che abbiamo introdotto in ambiente "falegnameria". Come abbiamo gia' osservato, prima di scrivere un programma conviene fare un esame approfondito del problema, in modo da averne presenti tutte le possibili articolazioni, i legami e le interrelazioni tra sezioni diverse di calcolo, e cosi' via.

Una metodologia ben assestata e' quella di documentare l'itinerario logico delle varie fasi e i loro collegamenti mediante flow-chart o diagrammi a blocchi. Ne esistono di almeno due categorie: quello delle fasi e quello logico.

Nel diagramma delle fasi, come dice il nome, interessa soprattutto l'articolazione generale del programma e il collegamento tra le diverse fasi di elaborazione, con riferimento al piano temporale di intervento di ogni fase anche in rapporto con le attivita' richieste all'operatore.

Nel diagramma delle fasi queste vengono rappresentate graficamente in modo che l'attenzione sia concentrata su dove e su chi svolge l'azione, anziche' sul modo in cui viene eseguita. Potremmo dire che il diagramma delle fasi sta a un programma come la scenografia sta ad un'opera teatrale.

Nel diagramma logico, invece, interessa descrivere il dettaglio delle singole fasi: in esso, quindi, appaiono tutte le biforcazioni logiche e gli algoritmi all'interno della fase. Da qui si passa alla scrittura del programma vero e proprio.

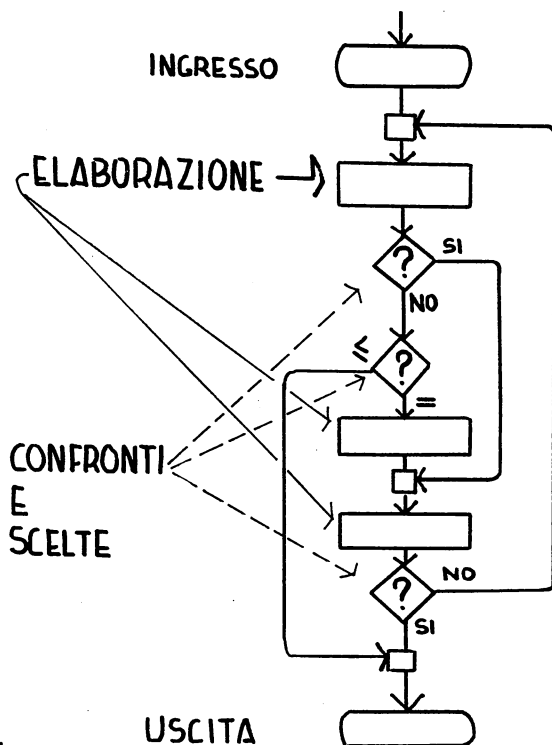



Figura 2.4

Naturalmente, non e' obbligatorio passare alla scrittura delle istruzioni di un programma attraverso la tecnica progettuale del flow chart; ne' esistono regole tassative per redigere un diagramma a blocchi, salvo qualche convenzione di carattere generale sui simboli grafici da adottare, come vedremo tra poco.

Si pensi che esistono programmatori che non solo saltano la stesura del diagramma a blocchi, ma addirittura passano dall'idea guida, appena abbozzata, direttamente alla digitazione delle istruzioni.

Per rappresentare le diverse operazioni in un diagramma a blocchi usiamo delle semplici notazioni grafiche, come quelle riportate in figura 2.4. Con tali simboli un algoritmo viene descritto in modo sintetico e verificato sul piano logico in modo molto piu' esauriente e chiaro che nel linguaggio naturale.

Una volta preparato il diagramma logico, la redazione di un programma dovrebbe risultare notevolmente piu' facile: e' sufficiente, infatti, sviluppare ogni simbolo nell'equivalente insieme di istruzioni del liguaggio prescelto.

La successione delle operazioni e' indicata dall'orientamento delle linee che collegano i vari simboli. Dalla figura appare altresi' chiaro che l'esito di un confronto puo' operare la selezione o il salto di operazioni.

2.2 COME SI SCRIVE UN PROGRAMMA

Un programma soddisfa l'idea-guida mediante una successione ordinata di istruzioni e produce l'effetto desiderato solo se eseguito per intero.

L'ordine di tale esecuzione viene dato dall'operatore, e ha effetto solo se il programma e' in memoria principale. In tali condizioni puo' essere ripetuto quante volte vogliamo. La parola chiave per dare l'ordine di esecuzione e' [RUN].

Al contrario del modo immediato e isolato di eseguire ogni singola istruzione indipendentemente dalle altre (si veda al riguardo quanto detto al paragrafo 1.4.1), un programma tratta globalmente un insieme di istruzioni che tra loro, l'ambiente interno della macchina e quello delle attivita' esterne dell'operatore danno vita a un processo definito di cause e di effetti piu' complesso del precedente.

Affinche' tale processo possa aver luogo e non dia risultati imprevedibili, bisogna scrivere il programma in modo che M24 sappia **quando** ogni istruzione deve "entrare in scena". La stessa definizione di programma indica che la successione delle istruzioni deve essere ordinata: bastera' mettere in testa a ogni istruzione un numero che rappresenti l'ordine di entrata.

Facciamo subito un esempio elementare, utilizzando due istruzioni fondamentali del Basic: l'istruzione di **assegnazione** e l'istruzione che consente di ottenere su video il valore di una variabile. Riprendiamo quindi l'esempio della paga trattato al precedente paragrafo.

Dopo aver accertato di essere in ambiente Basic controllando la presenza sul video del segnale [Ok], introduciamo da tastiera le corrispondenti istruzioni.

Piu' precisamente:

```
Ok

10▽L=200     [CR]
20▽PRINT▽L   [CR]

RUN          [CR]
```

Il simbolo ▽ sta per spazio; [CR] sta per operazione di chiusura del messaggio.

Commentiamo brevemente questo programma elementare e nello stesso tempo cominciamo a individuare le prime regole di scrittura del linguaggio. La prima di queste e' che ogni istruzione e' contrassegnata da un numero d'ordine, detto **numero di linea.** Nell'esempio il programma e' costituito da due sole istruzioni: la numero 10 che genera la variabile L alla quale l'istruzione assegna il valore 200; la numero 20 la cui funzione espressa dal verbo inglese PRINT (che significa stampare) legge dalla memoria il valore della variabile L e lo fara' apparire sul video. Quanto fin qui commentato e' solo sulla carta, anzi nella memoria di M24.

Per provocare realmente l'effetto desiderato, non basta introdurre le istruzioni nella memoria, ma bisogna anche introdurre un particolare comando per l'esecuzione del programma. Si tratta, come gia' prima accennato, del comando [RUN] che alla lettera significa [CORRERE], girare, e che traduciamo col termine [eseguire].

Nel programma -e' opportuno ripeterlo- ogni istruzione e' stata preceduta da un numero di sequenza per i seguenti due motivi.

1. Perche' vogliamo che M24 la prenda in carico, memorizzandola ma senza eseguirla subito.
2. Perche' vogliamo che abbia un ordine di esecuzione rispetto alle altre istruzioni.

La prima motivazione e' importante perche', in assenza del numero di linea, battendo [CR] l'istruzione verrebbe subito eseguita e, poiche' M24 non l'ha registrata, quando desiderassimo eseguirla ancora dovremmo digitarla tutta da capo.

Il secondo motivo e' anch'esso essenziale perche', in caso di assenza di tale numero e qualora M24 fosse stato istruito a eseguire il programma prendendo in esame le singole istruzioni secondo l'ordine con il quale sono state introdotte da tastiera, saremmo in seria difficolta' volendo saltare da una istruzione all'altra senza un elemento a cui riferirci. In pratica potremmo fare solo programmi lineari, tipo il rullo della pianola.

L'ordine con cui l'interprete del Basic prende in carico ed esegue le istruzioni di un dato programma in memoria e' quello secondo la succesione crescente dei numeri di linea. Al termine dell'esecuzione questa sequenza di istruzioni viene conservata nella memoria centrale, in attesa che M24, per nostro comando, la elabori nuovamente o la tratti diversamente.

Tra le regole di scrittura di un programma in Basic, oltre a quella che ogni linea deve iniziare con un numero, c'e' anche quella che, se

scriviamo due istruzioni con lo stesso numero, e' quella scritta per ultima che soppianta l'altra. Cosi' questa regola ci suggerisce il modo per cancellare un'istruzione: basta scrivere un numero uguale a quello della linea che contiene l'istruzione da cancellare. Dopo il numero si deve comunque battere [CR], altrimenti non arriva all'interprete.

Un'altra interessante proprieta' offerta dal linguaggio Basic e' quella di poter introdurre altre linee tra due contigue gia' registrate, purche' le nuove siano contrassegnate da numeri interi interni al campo di inserimento.

Se non fosse possibile a causa dell'esiguita' dei numeri disponibili in tale campo, e' sufficiente battere **[RENUM]** [CR] e tutte le linee di istruzioni registrate fino a quel momento vengono automaticamente rinumerate con intervallo 10 (se non altrimenti specificato). In tal modo si crea sempre un intervallo utile per i successivi inserimenti.

Per semplificare la scrittura di un programma esiste addirittura la possibilita' di far fare alla macchina la numerazione automatica delle linee.

Infatti, digitando il comando [**AUTO**] [CR], il Basic a ogni chiusura di linea [CR] si posiziona nella linea successiva a cui attribuisce un numero incrementato di 10 rispetto alla precedente.

Naturalmente, esistono varianti al comando AUTO per far cominciare la numerazione automatica solo da un certo numero e per variare il passo di numerazione, ma per semplicita' non li riportiamo in questa sede.

Per sbloccare l'automatismo della numerazione automatica, perche' altrimenti l'eventuale comando RUN di esecuzione verrebbe inglobato anch'esso nel corpo di una linea di programma, occorre digitare insieme i tasti [CTRL] e [CR].

Altri importanti comandi da utilizzare sono i comandi **LIST** e **EDIT.** Di quest'ultimo tratteremo nel prossimo paragrafo dedicato alle correzioni di un programma gia' registrato.

IL comando LIST consente di visualizzare il programma che M24 ha in memoria in un dato momento.

Per inciso, la memoria centrale puo' contenere solo un programma alla volta.

Il comando LIST e' di estrema utilita': infatti supponete che il programma appena eseguito contenga in testa l'istruzione CLS che pulisce il video da ogni informazione. Evidentemente scompare anche il programma che avevate digitato. Se volete rivederlo interamente di nuovo serve appunto il comando LIST.

Esistono varianti speciali di questo comando, qualora non desiderassimo visualizzare tutte le istruzioni del programma in memoria, ma solo una istruzione in particolare, oppure le istruzioni da...a..., oppure dall'inizio a..., oppure ancora, da...fino alla fine.
La sintassi e' la seguente, con riferimento a un programma fatto, ad esempio, di 10 istruzioni numerate da 10 a 100, di 10 in 10 (si veda la figura 2.5).

Per visualizzare le 4 istruzioni da 20 a 50:

```
LIST  20-50    [CR]
```

Figura 2.5

Figura 2.6

Per visualizzarne 5 dall'inizio fino alla 50 compresa, oppure dalla 50 fino alla fine, occorre rispettivamente battere:

```
LIST -50     [CR]   ,    oppure

LIST 50-     [CR]   .
```

Impariamo adesso l'uso di: **NEW, CONT, LLIST.** Il comando **NEW** consente di cancellare il programma presente in memoria centrale di M24 e svuotare il contenuto di tutte le variabili. Questo comando quindi fa "tabula rasa" in memoria e dopo la sua esecuzione non dovrete piu' preoccuparvi di nomi eventualmente gia' utilizzati da altri programmi: volendo, potrete riutilizzarli (solo i nomi) per le variabili che state per introdurre. Con il comando **CONT** e' possibile far riprendere l'esecuzione di un programma che fosse stato bloccato da una manovra di interruzione esterna (tipo CTRL+C o CTRL+SCROLL LOCK), oppure interna al programma con un'istruzione STOP, inserita in una certa linea. Il comando **LLIST** (con doppia L) consente infine di ottenere la lista del programma in memoria con uscita sulla stampante anziche' su video. Ovviamente la stampante deve essere collegata, accesa, con un foglio gia' inserito. Nella figura 2.6 sono riepilogati i comandi Basic fin qui trattati per la scrittura di un programma. Ancora qualche dettaglio sul comando RENUM. La variante piu' ricca di questo comando consente di rinumerare la successione delle istruzioni introdotte in memoria con un passo prescelto a partire da un dato numero di linea alla quale viene assegnata una nuova origine.

Figura 2.7

Ad esempio, scrivendo

```
RENUM 150,50,20     [CR]
```

tutte le linee a partire dalla 50 -che assumera' il nuovo numero d'ordine 150- in avanti verranno scaglionate di 20 in 20. Nell'istruzione soprastante avrete notato le virgole come separatori logici.

La variante piu rapida e' quella che gia' conoscete costituita dalla sola parola chiave RENUM, nella quale e' implicita la specifica di scaglionare **tutti** i numeri di linea di 10 in 10.

Uno degli esempi piu' interessanti, per gli scopi che saranno piu' chiari quando tratteremo come fondere i programmi tra loro, e' il seguente:

```
RENUM 320   [CR]
```

dove tutte le linee del programma in memoria sono coinvolte e viene fissato solo il numero di linea che costituisce la nuova origine assoluta del programma (in questo esempio 320), restando impliciti la precedente origine -che viene soppressa- e l'intervallo che e' lo standard di 10 in 10. Nella figura 2.7 riportiamo un compendio di quattro casi di rinumerazione prima e dopo l'esecuzione delle corrispondenti RENUM.

Riassumendo, con alcune ulteriori precisazioni:

- appena digitato [CR], dopo aver introdotto in memoria l'istruzione, l'interprete Basic esamina se questa e' stata etichettata con un numero di linea. In tal caso non l'esegue immediatamente e ne rinvia l'esecuzione finche' non riceve il comando globale [RUN],
- se, invece, l'istruzione e' priva del numero di linea, la considera un comando da eseguire immediatamente e subito dopo l'esecuzione viene cancellata dalla memoria.
- la fine dell'esecuzione di un comando viene segnalata dall'[Ok] sul video,
- il numero di linea puo' variare da 0 a 65529 e deve essere un numero intero,
- la capacita' massima di caratteri che una linea di programma puo' contenere e' 255, compresi quelli di servizio impiegati per indicare lo stesso numero di linea e per lo spazio di separazione con il corpo della prima istruzione,
- in una linea di programma e' possibile inserire piu' istruzioni divise dal separatore [:] finche' non si raggiunge la capacita' massima della linea di 255 caratteri, raggiunta la quale viene generato un [CR] automatico,
- quando cercate di rifare da capo un programma con la numerazione automatica compare un asterisco (*) dopo il numero di linea a indicare che avete ancora il vecchio programma in memoria, le cui istruzioni rischiate di cancellare appena batterete [CR],
- le linee di programma vengono eseguite per numeri di linea ascendenti, anche se sono state introdotte in memoria in ordine sparso.

LE PIU' COMUNI SITUAZIONI DI ERRORE

In fase di scrittura del programma possono verificarsi delle situazioni di errore che l'interprete Basic rivela con opportuni commenti di segnalazione sul video, immediatamente dopo il loro verificarsi. Tipiche situazioni di errore sono quelle rivelate a seguito di:

- introduzione del **solo** numero di linea che non sia gia' stato utilizzato per una precedente istruzione; il commento in questo caso sara' "INDEFINED LINE NUMBER" = numero di linea che non definisce alcuna istruzione,
- il mancato rispetto dell'ortografia del Basic, parole chiave scritte male, separatori fuori posto o errati, dimenticanza dello **spazio tra il numero di linea e la parola chiave**; in tutti questi casi l'unico laconico messaggio e' "SYNTAX ERROR".
- se avete erroneamente battuto un numero di linea in un certo modo, per esempio 36 355 Print il Basic non l'accetta in quanto dopo il numero linea 36 non c'e' una parola chiave ma il numero 355; in tali condizioni sopprime lo spazio e alloca la parola Print alla linea 36355. Per rimettere le cose a posto bisogna impostare nuovamente la linea fasulla e poi battere [CR]: in tal modo avete distrutto la linea 36355 e potete ribattere la linea.

2.3 COME SI CORREGGE UN PROGRAMMA

Una delle fasi piu' delicate e sfortunatamente piu' lunghe della programmazione e' quella dell'eliminazione degli errori formali, logici e di stile di un programma, ovvero la rimozione di tutte quelle cause che provocano effetti indesiderati o diversi da quelli che avevate progettato di ottenere. La messa a punto di un programma (in inglese debugging) si realizza mediante una serie di prove effettuate sul programma o parti di esso. Si introducono dei dati nella macchina e i risultati ottenuti vengono confrontati con quelli elaborati a mano.

Per un debugging serio occorre eseguire un insieme di prove -detto giocoprova- che contempli tutti o buona parte dei piu' significativi eventi che possono presentarsi in pratica nell'esercizio reale del programma.

La correzione di un programma, ovviamente, puo' avvenire solo se il programma e' in memoria. La correzione normalmente viene fatta a seguito del verificarsi di almeno una delle seguenti circostanze:

a. durante la scrittura di un programma a seguito di errore segnalato dall'interprete subito dopo la registrazione di una linea, oppure
b. durante la scrittura di un programma a seguito di errore che il programmatore coglie prima di registrare la linea in memoria, oppure
c. durante il tentativo di far eseguire il programma.

Nel secondo caso sta al programmatore ricondursi al primo caso concludendo la linea con [CR] e poi valutare se convenga procedere alla correzione, oppure cancellare l'intera linea e ricominciare da capo.

La correzione di una linea gia' registrata in memoria richiede invece di passare allo stato **EDIT** -un sottoambiente del Basic- mediante il corrispondente comando. La correzione puo' consistere nella modifica di uno o piu' caratteri da aggiungere, sostituire o cancellare nella linea di istruzione segnalata dall'errore.

Per queste modifiche e' opportuno richiamare le funzioni di certi tasti: alcuni di questi sono gia' noti, come quelle dei tasti [←] o backspace e il [DEL], che devono essere premuti in accordo con la posizione del cursore che, come un mirino, traguarda il carattere da cancellare.

Talvolta, pero', la correzione non consiste nel cancellare ma nell'inserire uno o piu' caratteri: a tal scopo provvede il tasto [INS], che fa passare appunto la tastiera allo stato "Inserzione". Va notato che dal momento che tale tasto e' in condominio con la cifra 0 (zero) della tastiera numerica occorre verificare quale delle due funzioni e' attiva. Per far cio' e' sufficiente controllare se la lampadina rossa incorporata nel tasto NUM LOCK e' accesa. Qualora lo fosse va spenta premendo il suddetto tasto. Finalmente possiamo vedere all'azione il tasto [INS].

La richiesta di passaggio allo stato EDIT presuppone di essere gia' in ambiente Basic (segnale di **Ok** presente).

Il passaggio avviene digitando:

EDIT xxxxx [CR]

dove xxxxx e' il numero di linea in cui vogliamo fare la correzione.

La risposta dell'interprete Basic e' immediata. Compare infatti sul video la corrispondente linea del programma contrassegnata da quel numero, la cui cifra piu' significativa e' sottolineata dal cursore. Premendo piu' volte il tasto [→] (quello della tastiera numerica che ha sul cappuccio il "6") facciamo scorrere il cursore verso destra. Tale scorrimento e' arrestato dal programmatore quando l'errore viene localizzato. A questo punto, traguardata la posizione da correggere, si puo' procedere in vari modi. Riferendoci a un esempio concreto, supponiamo di avere in memoria alla linea 100 la seguente istruzione:

a) 100 PRINT A,ELBA

che vogliamo correggere in modo che diventi:

b) 100 PRINT A,B,ALBA;

Battendo il comando EDIT 100 [CR] sul video apparira':

c) 100 PRINT A,ELBA

La possibile procedura di correzione e' quella indicata nei seguenti passi:

1. con il tasto [→] ci spostiamo a destra finche' il cursore non sta sotto la "E" di ELBA. La situazione si presenta cosi':

```
100 PRINT A,ELBA
```

2. entriamo in stato "Inserzione" premendo il tasto [INS], facendo attenzione che sia spenta la lampadina del tasto [SCROLL LOCK], altrimenti invece di entrare in stato "INSERZIONE" sostituiamo un "0" al posto della "E" di ELBA.
   Se la manovra e' riuscita ce ne accorgiamo perche' la forma del cursore e' cambiata: la lineetta si e' trasformata in un cuneo, o in un quadrato, a seconda della densita' di carattere in cui il video si trova in quel momento: comunque sia, in stato inserzione, qualunque tasto digitiamo provoca l'inserimento del corrispondente carattere stampabile a sinistra del cursore. Cosi' ora battendo i tasti [B] e [,] otteniamo l'inserimento dei corrispondenti simboli e la situazione al video sara' la seguente:

```
100 PRINT A,B,ELBA
```

   (Il cursore e' ancora in stato inserzione).
   Per correggere ELBA in ALBA possiamo farlo almeno in due modi:

   . inserire una A a sinistra della E; in tal modo il cursore si sposta traguardando la E di ELBA che puo' essere cancellata battendo il tasto [DEL]; oppure,

   . uscire dallo stato "Inserzione", ribattendo il tasto [INS], o spostando il cursore con i tasti delle frecce; in tal modo il cursore riassume la forma normale. Traguardando la A di ALBA si batta ora il tasto [E] il cui corrispondente carattere si sostituira' alla A.

   Dopo aver fatto la correzione in uno dei due modi sopra indicati, bisogna ricordarsi di confermarla premendo il tasto [CR]: in tal modo la linea di programma richiamata con il comando EDIT va veramente a sostituirsi con quella originaria del programma di cui faceva parte.

Quanto fin qui indicato potrebbe gia' bastare se la correzione riguardasse una sola linea di programma; normalmente le correzioni si apportano avendo un gruppo di linee sotto osservazione sul video.
Possiamo indirizzarci alla linea che ci interessa semplicemente muovendo il cursore, in quanto l'ambiente Edit di M24 puo' estendersi a tutte le linee di istruzioni Basic che il video riesce a contenere in un dato momento: cosi' il recupero della posizione errata e la relativa correzione puo' essere condotta muovendo il cursore in tutti i modi possibili e con i tipi di spostamenti che i tasti direzionali consentono.
Non e' naturalmente possibile in questo libro esaminare tutte le possibili strategie che conducono il cursore alla posizione errata. Ciascuno trovera' il "suo" metodo personale per far arrivare il

cursore in "zona errore".

La tecnica per far le correzioni verra' poi condotta in uno dei modi prima segnalati, facenti uso dei tasti [INS], [DEL] o [←], o sostituendo direttamente i caratteri nuovi sopra quelli errati.

Come gia' prima specificato, dopo ogni correzione apportata a una certa linea di programma, occorre convalidarla premendo il tasto [CR], prima di spostare il cursore nella zona errore di un'altra linea. In caso contrario le correzioni apportate riguarderanno solo la memoria associata al video ma non quella in cui viene conservato il programma e che viene interessata tutte le volte che battiamo il comando "RUN". Sara' quindi opportuno accertarsi del buon esito delle correzioni apportate lanciando il comando LIST che ripresentera' il programma prelevandolo dalla memoria di lavoro.

2.4 COME SI USA UN PROGRAMMA

Quanto qui tratteremo e' gia' stato in buona parte introdotto nei precedenti paragrafi. Rimane invece da considerare il ventaglio delle possibilita' d'impiego in un programma di tutte le risorse che M24 ha a disposizione. Vedremo quindi come dal complesso scenario di questi mezzi possiamo assimilare le caratteristiche essenziali per migliorare la progettazione di un programma.

Innanzi ricordiamo che e' possibile durante l'esecuzione di un programma che sta fornendo certi risultati su video (numeri o grafici), riprodurre in stampa la copia fedele di quanto sia stato fino a un certo momento visualizzato. Come gia' sapete si usa dire: "ottenere l'**hard-copy** (copia materiale) del video". Per ottenere questo risultato e' sufficiente eseguire la manovra [↑] + [PrtSc] avendo avuto cura di accertarsi preventivamente che la stampante grafica sia opportunamente predisposta (collegata all'M24, accesa e con un foglio inserito).

Se il video contiene anche dei grafici la loro riproduzione su carta avviene se prima di passare in **gwbasic** si e' richiamato dal sistema MS-OS in memoria di lavoro il componente **graphics**.

Descriviamo adesso le principali fasi organizzative che interessano l'uso di un programma. La figura 2.9 illustra il "ragno" delle concatenazioni possibili tra i due principali ambienti e nell'ambito di ciascuno di essi i rispettivi comandi di competenza, maggiormente impiegati.

Tra questi potrete notare i comandi che riguardano il traffico delle comunicazioni tra memoria centrale e dischetti, cioe', in Basic, i comandi LOAD (carica) e SAVE (salva). La funzione e' richiamata dal loro stesso nome. Con la parola LOAD, infatti, si attiva un'operazione di trasferimento di dati da dischetto verso la memoria centrale, mentre con SAVE si ottiene l'operazione contraria di trasferimento da memoria a dischetto.

Ovviamente, non basta solo la parola: nell'ambito del comando, occorre anche specificare il nome dell'insieme dei dati (file), oggetto del trasferimento e quali dei due dischetti coinvolgere nell'operazione. Tutti questi elementi vanno dichiarati nel rispetto di una certa punteggiatura.

copy
type
date
time
rem
dir
erase
chkdsk
diskcopy
graphics
format
MS-DOS
(>)
comp
mode
PR 15 B
sys
pause
gwbasic
system
list
shell"
edit
BASIC
save
OK
OK
clear
load
files"
new
run
renum
BREAK
CTRL
C
load
save
CPU
PROGRAMMA
MEMORIA CENTRALE

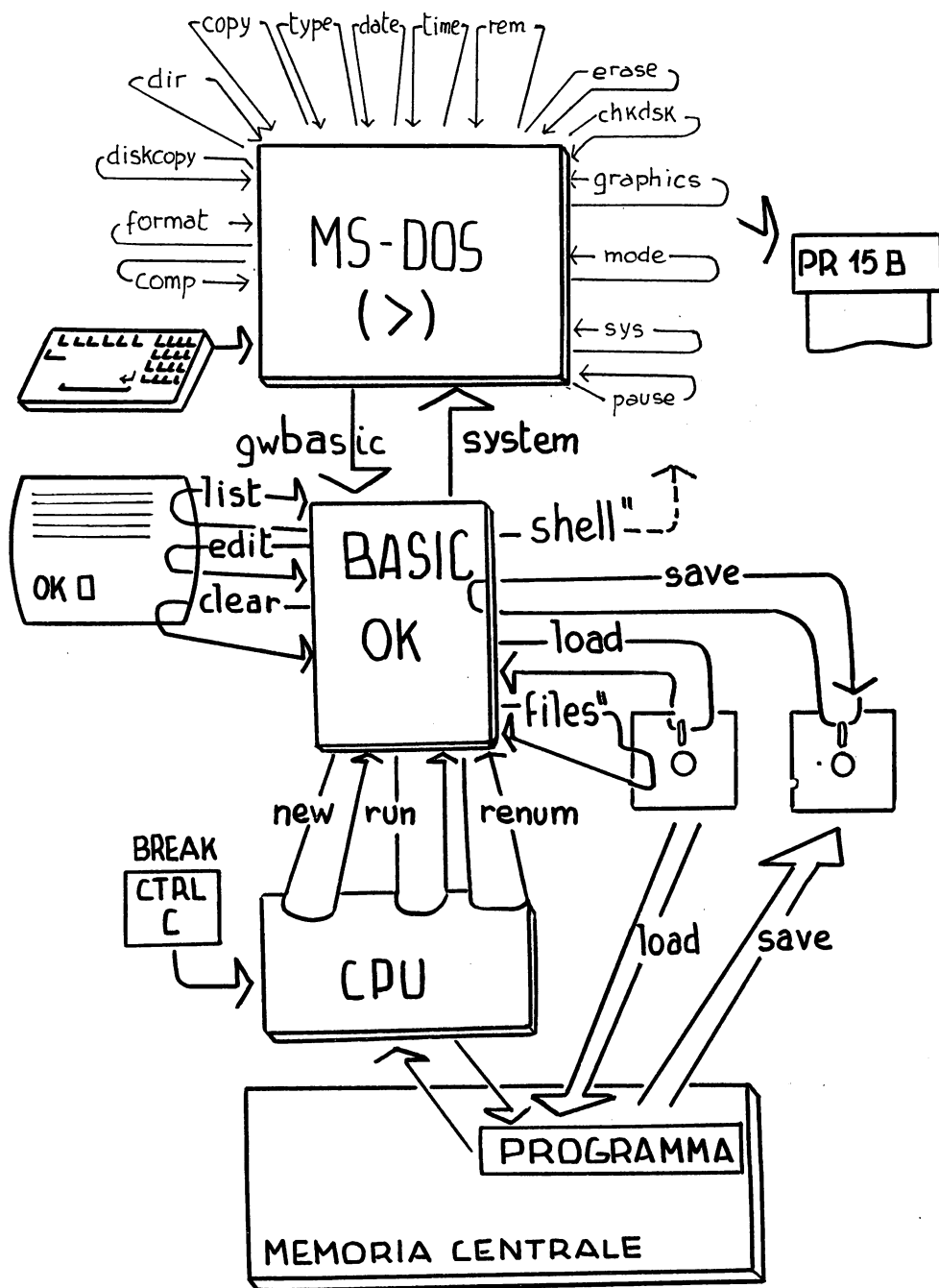



Figura 2.8

Per esempio se volete mettere in salvo sul dischetto **b** (quello superiore) un programma appena scritto e ancora residente in memoria centrale, dovrete scegliergli un nome evitando quei pochi non ammessi (vedi oltre) come le parole chiave del Basic -tipo PRINT, RUN, LIST, EDIT, etc...- per non creare ambiguita' a livello Interprete.
Sia, ad esempio, [alfa] il nome prescelto; il comando sara' allora:

```
SAVE"b:alfa" [CR]
```

Fate attenzione che uno o piu' [spazi] prima o dentro il nome vengono interpretati in modo ambiguo da gwbasic. Se per esempio digitate:

```
SAVE"b:▽alfa"   [CR]   oppure
SAVE"b:alfa▽"   [CR]   oppure
SAVE"b:al ▽ fa" [CR]   etc.
```

la risposta dell'interprete sara' nel primo caso TOO MANY FILES, che significa: (hai assegnato) troppi nomi di file (al tuo programma da salvare).

Il nome del file viene accolto favorevolmente dall'interprete negli altri due casi ma se andate a verificare come sono stati registrati i loro nomi nell'indice (directory) del dischetto (vedi oltre) vedrete che appare solo quanto dichiarato prima degli spazi (nel caso specifico "al").

Durante l'operazione di registrazione del programma dalla memoria i lavoro al dischetto, vedrete accedersi la lampadina rossa accanto alla fessura del drive b: significa che il trasferimento e' in atto. Appena si spegne appare [Ok] sul video per segnalare che l'operazione e' conclusa e che il Basic e' nuovamente a vostra disposizione. Intanto il vostro programma, pur essendo ancora in memoria, e' stato messo in salvo col nome assegnato sul dischetto b: se va via la corrente o spegnete M24, potrete riottenerlo in memoria col comando complementare [LOAD].

```
LOAD"b:Alfa"[CR]
```

Una liberta' ammessa nella dichiarazione dei nomi e' la scrittura maiuscola o minuscola: in ogni caso la designazione del nome, sia in fase di ricerca, sia in fase di registrazione viene fatta con lettere maiuscole. Quando non si ricorda l'esatta dizione del nome con cui un file e' stato registrato (al limite per recuperare un nome dimenticato) basta digitare in ambiente Basic :

```
FILES"b:" [CR]
```

e appariranno su video **tutti** i nomi dei file registrati sul dischetto inserito nel drive b.

Analogamente per i file registrati sul dischetto inserito nel drive a (inferiore); in quest'ultimo caso, naturalmente, il comando dovra' essere impartito cosi':

```
FILES"a:"[CR]
```

o piu' semplicemente

FILES [CR]

se il drive selezionato implicitamente da M24 (drive di default) e' proprio il drive a.

Si noti che per richiamare i programmi registrati con nomi che hanno un coprpo interno di spazi (nel casi sopra riportati con uno o due spazi), mentre nella lista dei nomi appaiono entrambi sotto il nome ("al"), occorrera' farlo ricordando esattamente il numero degli spazi e inoltre aggiungendo in coda il resto del nome.

Puo' quindi essere un modo per impedire di richiamare un programma da parte di chi non conosce questa chiave segreta espressa in numero di spazi e una coda. Altri comandi utili sono NAME e KILL. Con tali comandi, come sara' meglio detto nel capitolo che tratta la gestione dei files, e' necessario specificare, oltre al nome del file, anche l'estensione .bas in modo che il gwbasic sappia che si tratta di un programma.

Con il comando **name** si ridefinisce il nome di un file gia' registrato in memoria. Se vogliamo, nell'esempio precedente, cambiare il nome [alfa] in [beta], basta digitare:

NAME"b:alfa.bas"as"beta.bas" [CR] (significa alfa **come** beta)

e il nuovo nome del programma diventera' [beta].

Il comando KILL (uccidi) cancella dal dischetto il file programma il cui nome viene specificato nel comando. Se, ad esempio, digitiamo:

KILL"b:beta.bas"[CR]

non troveremo piu' il file [beta.bas]. La verifica puo' essere eseguita con il comando FILES gia' visto.

PROGRAMMARE USANDO I TASTI FUNZIONALI

Le parole chiave dei comandi fin qui usati sono state associate da **gwbasic** ad alcuni tasti funzionali. In particolare battendo F1, F2, F3, F4 e' come se battessimo LIST , RUN+[CR], LOAD ,SAVE . In tal modo si risparmia tempo tutte le volte che vogliamo listare le istruzioni del programma registrato in memoria. Altrettanto dicasi per lanciarlo in esecuzione: anzi in tal caso al tasto F2 oltre al comando RUN seguito da uno [spazio] e' stato accoppiato anche il codice generato dalla pressione del tasto [CR].

Ai tasti F3 e F4 occorre, ovviamente, far seguire il nome del programma da caricare o da salvare. Le relative operazioni interessano, salvo diverse indicazioni, il dischetto presente nel drive di default.

Nella riga 25 del video **gwbasic** presenta le parole accoppiate ai tasti funzionali: tra questi F9 riguarda appunto la scelta di conservare o meno tale riga 25 alla visualizzazione delle parole accoppiate. Infatti, battendo F9 OFF e' come se battessimo KEY OFF, che tradotta in linguaggio comune significa: rendere libera la riga 25 del

video. Per rioccuparla con l'indicazione delle parole accoppiate basta battere F9 ON.

I tasti F7 e F8 sono complementari: F7 attiva la funzione di TRACE, F8 la disattiva. Brevemente, la funzione di TRACE consente al programmatore di avere la traccia dei percorsi, espressi dai numeri di linea interessati, durante l'esecuzione di un programma.

Ulteriore risparmio di tempo si puo' fare usando in accoppiata il **tasto speciale** [ALT] con le iniziali di alcuni dei verbi del vocabolario BASIC. Ad esempio per battere AUTO, INPUT o PRINT basta battere, rispettivamente A, I o P tenendo premuto il tasto [ALT].

2.5 UN ESEMPIO INTRODUTTIVO

Cerchiamo ora di mettere in pratica quanto fin qui appreso con la scrittura di un semplice programma applicativo.

Come idea-guida prendiamo un problema che si presenta abitualmente nella progettazione grafica di un testo. In questo genere di problemi se avete gia' fissato quante battute far stare in una pagina, generalmente vi chiedete:

"Come organizziamo lo spazio utile della pagina? - Quante righe? Quanti caratteri (battute) per riga?"

Orbene, partendo da queste semplici domande passiamo a una prima formalizzazione del problema e stabiliamo anche un po' di terminologia.

Tanto per non inventare vocaboli nuovi, impieghiamo quelli che usano realmente i grafici per questo problema. Cio' che risponde ai nostri quesiti e' la determinazione delle dimensioni di quello spazio di cui prima dicevamo, ovvero della cosiddetta "gabbia". Geometricamente parlando il problema consiste nella ricerca dei valori dei lati di un certo rettangolo in cui imprigionare il testo.

"Ma come verra' stampato il testo dentro la gabbia? Con quale misura di caratteri? Quanto spazio tra carattere e carattere, e tra riga e riga?"

Le caratteristiche della scrittura si esprimono appunto in spaziature orizzontale e verticale, che altro non sono che densita' lineari.

Cosi' il problema puo' essere enunciato nel seguente modo:
"Dato il numero totale di caratteri della pagina (CP), calcolare le dimensioni della gabbia (LG e HG), cioe' lo spazio utile per la stampa al variare delle densita' orizzontale (DO) e verticale (DV) e del numero totale delle righe per pagina (RP)".

Con riferimento alla figura 2.9 , nella quale sono rappresentate le grandezze sopra definite, vediamo che il problema non e' classificabile tra quelli piu' semplici, come il calcolo dell'area di un rettangolo date le misure dei lati.

Nel nostro caso assumendo come dati di partenza il numero totale CP dei caratteri della pagina standard ci proponiamo di esaminare come variano HG e LG, immettendo nella procedura di calcolo -che dobbiamo peraltro ancora precisare- una terna di valori da assegnare rispettivamente a DO, DV e RP.

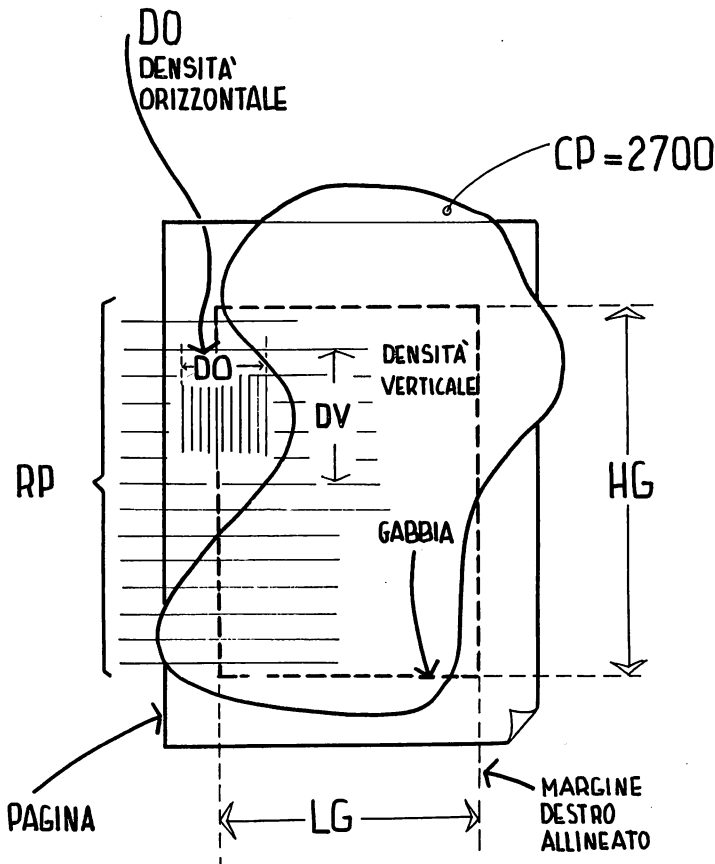



Figura 2.9

In pratica vogliamo fabbricare un algoritmo per mezzo del quale sperimentare diverse terne di valori e valutare criticamente, con sensibilita' grafica, i corrispondenti valori di HG e LG espressi in millimetri, che individuano la gabbia. S'intende, implicitamente, che la stampa del testo debba avvenire col perfetto incolonnamento del margine destro delle righe.

Organizziamo ora le nostre idee per costruire l'algoritmo, cercando di legare tra loro i concetti sopra indicati, e le variabili che vi stanno sotto.

Primo passo. Come si esprimono i concetti di densita'. Dalle definizioni di densita' date negli enunciati, facciamo osservare che mentre DO e DV si riferiscono rispettivamente ai numeri di caratteri e di righe per unita' di lunghezza (per esempio 1 cm), RP, pur essendo ancora una densita', esprime invece quante righe stanno in una pagina. Quindi, al contrario di DV che e' una densita' verticale lineare (righe al cm), RP e' una densita' riferita a un'area (righe per pagina).

Secondo passo. Da quanto detto sopra discende la definizione del numero di caratteri per "riga standard", che designeremo col simbolo CR. Tale grandezza esprime in altro modo la densita' orizzontale.
La differenza tra DO e CR e' che mentre la prima esprime la densita' per cm, la seconda fornisce il numero dei caratteri contenuti nella riga standard. La cosa notevole del discorso e' che CR risulta legato e percio' dipendente dai valori che assegniamo ai simboli CP e RP. Sarebbe concettualmente errato assegnare a CR un valore arbitrario e indipendente, come se fosse un dato esterno all'algoritmo, a cui assegnare un valore qualsiasi. Tale valore potrebbe non risultare compatibile con quelli che assegniamo a CP e a RP e l'algoritmo sarebbe ambiguo. Infatti il legame di dipendenza tra quelle tre variabili e' sintetizzato dalla formula:

CR = CP/RP

dove assegnando alle **variabili** CP e RP valori arbitrari (ma in un certo campo utile di manovra) si ottiene il corrispondente valore di CR. Poiche' il numero totale dei caratteri per pagina (CP) non vogliamo cambiarlo nel corso dei vari esperimenti, lo assumeremo costante e pari, ad esempio, a 2700 (caratteri per pagina). In tal caso questa condizione viene registrata dentro la formula che calcola CR, cioe':

CR = 2700/RP

Terzo passo. A questo punto bisogna cominciare a legare le dimensioni della gabbia -che, non dimentichiamo, materializzano la nostra idea guida- con le altre grandezze. La prima relazione utile e':

LG = CR/DO

per mezzo della quale la lunghezza della riga (o larghezza della gabbia) si ottiene calcolando il rapporto tra il numero di caratteri per riga e la densita' orizzontale espressa col numero di caratteri che stanno in un centimetro di riga.
Siccome vogliamo ottenere LG espresso in millimetri anziche' in centimetri, sara' necessario mettere a posto la formula in modo che il risultato sia 10 volte maggiore. Cosi' arriviamo alla formula definitiva:

LG = 10.CR/DO

Quarto passo: Analogamente, se leghiamo l'altra dimensione della gabbia (HG=altezza) con la densita' verticale, espressa in numero di righe al centimetro, si potra' scrivere:

HG = 10.RP/DV

Quinto passo: Stesura del diagramma logico per la verifica d'insieme. Con i dati del problema DV,DO e RP, riferiamoci alle figure 2.10 e 2.11 in cui sono illustrati due diversi modi di seguire l'esecuzione delle procedure che conducono alla determinazione di HG e LG. Nella figura 2.11 la procedura e' disegnata con l'intenzione di riprodurre, per quanto possibile, il processo che si svolge all'interno dell'elaboratore; parti di memoria, simili a scatole, vengono automaticamente attivate e designate col nome che abbiamo scelto di assegnare alle variabili dell'algoritmo.

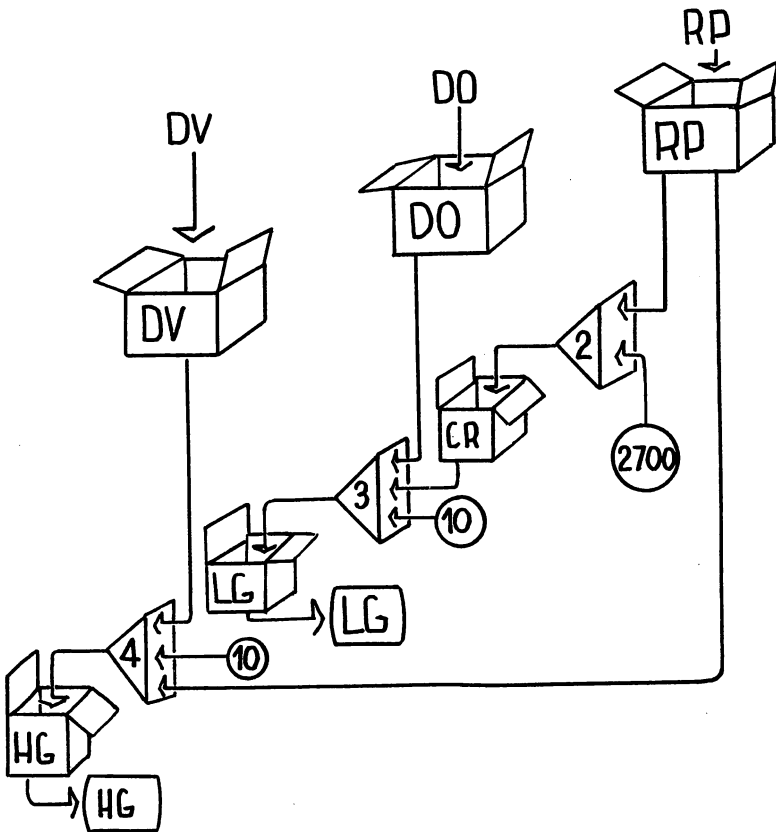



Figura 2.10

In figura 2.11 l'algoritmo viene invece descritto attraverso il diagramma logico col quale formalizziamo sia l'aspetto temporale della procedura sia i momenti di calcolo. A fianco, per comodita', e' anche riportata la descrizione del calcolo espressa in linguaggio naturale.

Sesto passo. Collaudo dell'algoritmo con dati di prova. Assumiamo DV=3, DO=5, RP=54 e assegniamo questi valori alle corrispondenti variabili nei diversi momenti di calcolo:

INIZIO
DV, DO, RP
2 CR=2700/RP
3 LG=10.CR/DO
4 HG=10.RP/DV
LG, HG
FINE
Introduzione dei valori da assegnare alle variabili DV, DO e RP.
Si calcola CR dividendo 2700 (numero di caratteri per pagina) per RP (numero di righe per pagina).
Si calcola LG moltiplicando per 10 il rapporto tra CR e DO.
Si calcola HG moltiplicando per 10 il rapporto tra RP e DV.
Visualizzazione di HG e di LG.

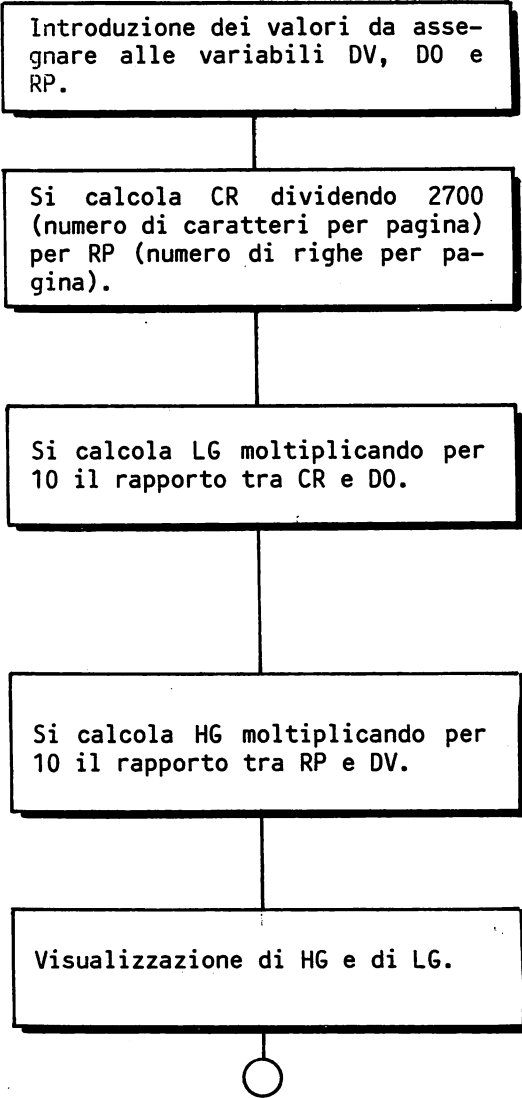
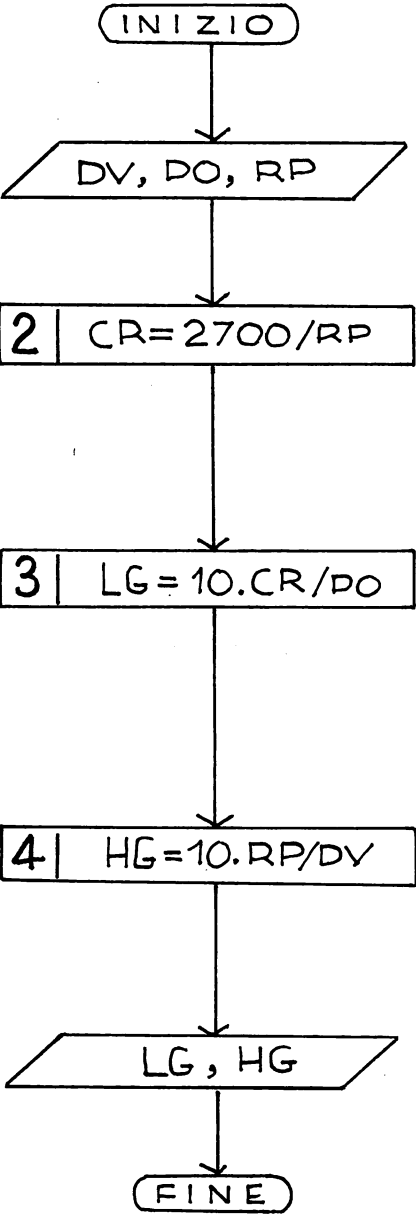



Figura 2.11

Momento 2. CR=2700/RP=2700/54=50 (come vedete a RP abbiamo assegnato il dato di prova 54)

Momento 3. LG=10.CR/DO=10.50/5=100 mm (il valore di CR e' preso dal risultato del momento precedente)

Momento 4. HG=10.RP/DV=10.54/3=180 mm

Dal diagramma logico al corrispondente programma scritto in Basic il passo e' abbastanza breve. Ricordando quanto gia' detto al paragrafo 2.1 a proposito delle istruzioni di assegnazione, aggiungiamo qui altre possibilita' offerte da queste istruzioni. In esse, infatti e' anche possibile assegnare alla variabile che compare alla sinistra del segno di uguaglianza -e che abbiamo chiamato convenzionalmente **primo membro** per analogia con le espressioni algebriche- il risultato delle operazioni effettuate sulle variabili che compaiono al **secondo membro**. In tal modo, tornando all'esercizio della gabbia, l'operazione 2 (CR=2700/RP) si svolge secondo il seguente schema:

- estrarre da quel deposito di memoria designato col nome RP il valore **attuale** della variabile RP,

- eseguire la divisione tra 2700 e il valore di RP,

- assegnare il risultato della precedente divisione alla variabile CR, il cui deposito in memoria (che portera' lo stesso nome) viene automaticamente reso disponibile.

Avrete notato la qualifica **attuale** che abbiamo riservato al valore di RP. E' bene fin d'ora abituarsi all'idea di provvisorieta' dei valori che una variabile puo' assumere. Nel senso che ad essa possono essere assegnati valori diversi di volta in volta. Quello che interessa e' la nozione di **quando** ha un certo valore. Vedremo che queste sottigliezze hanno la massima importanza in quegli algoritmi ciclici in cui una stessa procedura puo' essere ripetuta piu' volte. In tali casi assume particolare rilievo sapere il valore che una certa variabile aveva nel precedente ciclo rispetto al valore attuale.

Non vorremmo troppo complicare le cose, soprattutto perche' la semplicita' del programma che abbiamo sviluppato come esempio introduttivo non sfrutta ancora appieno il concetto sopra esposto. Pero' una cosa possiamo tentare di ricordare per sempre: ogni volta che la macchina assegna un valore ad una variabile, questa perde automaticamente il suo precedente valore.

Richiamando l'analogia del recipiente, ogni volta che assegniamo un nuovo valore e lo introduciamo nella memoria contenitore con la semplice designazione del suo nome, il vecchio contenuto fuoriesce automaticamente. C'e' come un azzeramento preventivo della relativa memoria. Se vogliamo per qualche motivo salvare il vecchio valore, quindi, dovremo trasferirlo in un altro contenitore, che evidentemente dovremo chiamare in modo diverso per non essere confuso con quello che contiene il nuovo valore.

Indipendentemente da questa singolare gestione dei contenitori che si autosvuotano ogni volta che vi mettiamo dentro qualcosa, c'e' un altro importantissimo rilievo da fare sulle istruzioni di assegnazione: le notazioni che il linguaggio usa per esprimere l'operazione di assegnazione non rispettano quelle che abitualmente impieghiamo per le espressioni algebriche.

In altre parole, espressioni come **A = A + 1** che non rispettano il principio dell'eguaglianza dei due membri, sono assolutamente lecite in Basic. In tal caso l'istruzione aggiunge il valore 1 al valore attuale di A. Se ad esempio, prima dell'istruzione, il valore di A era 10, ad istruzione eseguita e' A=10+1.

Come conseguenza, nello scrivere o leggere le istruzioni di assegnazione converra' sempre **pensare prima al secondo membro** (quello alla destra del segno [=]) per prevedere gli effetti sul primo. Inoltre il primo membro non **dovra'** mai contenere piu' di una variabile e per giunta, da sola, nella sua forma piu' elementare. Pertanto espressioni come:

A+B=A   A/2=B   A-3=B   2^A=A   etc.

non sono ammesse e dovranno essere riscritte in forma equivalente dove A compare isolata al primo membro nella forma:

A= ...

Come avrete certamente compreso, la macchina non puo' direttamente operare su concetti matematici come fa l'uomo ma soltanto sui simboli che li rappresentano e per di piu', talvolta, usando convenzioni che in parte stravolgono le stesse notazioni matematiche.

Qui di seguito riportiamo alcuni esempi significativi di istruzioni di assegnazione:

1)	A=0	azzeramento della variabile A
2)	A=5	assegnazione del valore 5 alla variabile A
3)	A=Z	trasferimento del valore attuale della variabile Z nella variabile A
4)	A=B+C A=B/C A=B*C A=A^2	trasferimento nella variabile A del valore risultante da operazioni eseguite su variabili al secondo membro. Questo tipo di assegnazione potremmo chiamarlo di "assegnazione calcolata".

Nel nostro problema tipografico, trasformiamo ora le espressioni nei blocchi 2, 3 e 4 del diagramma logico di figura 2.11 nelle corrispondenti istruzioni di assegnazione Basic, tutte del tipo calcolato come sopra definito. Tenuto conto degli operatori aritmetici usati in Basic (si veda il paragrafo 1.4), le espressioni diventano:

```
20 CR=2700/RP
30 LG=10*CR/DO
40 HG=10*RP/DV
```

Rimangono da definire le istruzioni che creano i depositi delle variabili DV, DO e RP nei quali raccogliere i dati introdotti e quelle per portare all'esterno della macchina i risultati che M24 ha messo nei depositi LG e HG.

Per l'introduzione dei dati usiamo l'istruzione **INPUT**, che in inglese significa "**metti** (to put) **dentro** (in)". In tale istruzione e' contenuto il nome di una variabile. Quando la macchina incontra una INPUT si ferma in attesa che l'operatore introduca un valore da assegnare alla variabile.

Addirittura, in una stessa INPUT, separate dalla virgola, e' possibile dichiarare piu' di una variabile. Nel nostro caso, allora, bastera' scrivere:

```
INPUT DV,DO,RP.
```

Per l'uscita dei risultati, cioe' per visualizzare i valori delle variabili LG e HG, useremo invece l'istruzione **PRINT**, che in inglese significa "**stampa**".

Occorre, naturalmente, specificare i nomi delle variabili di cui vogliamo conoscere i valori. A tal fine scriveremo:

```
PRINT LG,HG
```

Finalmente il programma e' completo e risulta cosi' articolato:

```
10 INPUT DV,DO,RP
20 CR=2700/RP
30 LG=10*CR/DO
40 HG=10*RP/DV
50 PRINT LG,HG
```

Facciamo notare, ancora una volta, che ogni istruzione, per far parte di un programma, deve essere preceduta da un numero di linea. Per la registrazione di ogni linea di istruzione occorre battere il tasto [CR]. Scritto il programma e mandato in esecuzione con il comando [RUN], appare sul video un punto interrogativo [?]. Tale simbolo vi avverte che M24 sta aspettando che voi digitiate il valore della prima variabile, **seguito dalla virgola**; quindi il valore della seconda variabile, **seguito ancora da una virgola** e infine il valore della terza variabile, **seguito da [CR]**.

Sara' bene osservare che quella appena descritta e' l'unica prassi accettata dall'istruzione INPUT scritta in quella forma: qualunque cosa possa sembrare irrilevante per voi ma costituisca uno scostamento dalla regola provochera' la visualizzazione di un messaggio di invito a ricominciare tutto da capo (REDO FROM START).

Rispettando invece la grammatica del linguaggio i risultati saranno quelli sotto indicati; e, inoltre, ricordatevi che per ripetere il calcolo bisogna digitare ogni volta il comando [RUN] [CR] che vi fara' comparire ancora il segno [?]. Da qui nuova attesa di valori da parte di M24, e cosi' via.

```
10 INPUT DV,DO,RP
20 CR=2700/RP
30 LG=10*CR/DO
40 HG=10*RP/DV
50 PRINT LG,HG
Ok_
RUN
? 5,3,50
 180          100
Ok_
RUN
? 10,10,60
 45           60
Ok_
RUN
? 0,3,50
Division by zero
 180          1.701412E+38
Ok_
```

```
RUN
? 5,0,50
Division by zero
 1.701412E+38  100
Ok_
RUN
? 5,3,0
Division by zero
Overflow
 5.671373E+37  0
Ok_
run
? 0,0,0
Division by zero
Overflow
Division by zero
Division by zero
 1.701412E+38  1.701412E+38
Ok_
```

Altri inconvenienti possono accadere se vengono introdotte particolari terne in cui a una o piu' variabili si assegnano valori nulli.
Esempio: [0,3,50], oppure [5,0,50], oppure [5,0,50]. Tali valori,infatti, vengono assegnati ai denominatori delle frazioni scritte alle linee 20,30,40 e poiche' il risultato tenderebbe a valori infiniti, la macchina li rifiuta e al loro posto risponde con il messaggio [division by zero].
Se, in particolare, provate a dare valore nullo a tutte le variabili e quindi battete la terna [0,0,0], otterrete le seguenti risposte:

a) Division by zero
b) Overflow
c) Division by zero
d) Division by zero
e) 1.701412E+38   1.701412E+38

Il commento in a) si riferisce alla prima divisione impossibile (per zero) della linea 20 a causa di RP=0. Analogamente per i commenti in c) e in d) associati rispettivamente alle linee 30 e 40. Il commento in b) significa invece superamento delle capacita' di memorizzazione del deposito che la macchina destina alle variabili. Per la variabile CR infatti il rispettivo deposito non riesce a contenere il massimo numero (1.701412E+38=1701412 seguito da 32 zeri) che la macchina genera come risultato della divisione per zero. In e) i risultati "degeneri" per LG e HG corrispondenti alla terna [0,0,0].
Un altro inconveniente puo' accadere se, avendo dimenticato di lanciare in esecuzione il programma con la [RUN] battete dei numeri, ad esempio [3,5,60], e poi concludete con [CR]. Quando poi lancerete il programma con il RUN avrete la segnalazione di errore:

```
SYNTAX ERROR IN 3
 Ok
 3 ,5,60
```

Il motivo di questa segnalazione e' che l'interprete Basic non considera i valori come quelli da assegnare alle variabili DO,DV,RP, perche' la loro digitazione e' avvenuta in un **momento** sbagliato, quando la macchina non aveva ancora il programma in esecuzione. Se vi foste accorti, prima della loro digitazione, che sul video non c'era il segnale [?], ma [Ok], vi sareste trattenuti dal concludere l'introduzione con [CR] e avreste potuto cancellare i caratteri introdotti con manovre tipo [←] o [CTRL H]. Avete invece introdotto la terna [3,5,60] in presenza di [Ok] e quindi il Basic l'ha diversamente interpretata.

"Quale potra' essere stata l'interpretazione?" Andiamo per ordine. La chiave della risposta sta nel messaggio di errore fornito dalla macchina. Se lo analizzate bene, vi accorgerete che, oltre al laconico SYNTAX ERROR, si specifica anche dove e' stato trovato l'errore: [..in 3]. Allora e' chiaro che il programma contiene un'istruzione alla linea 3 e il processo che l'ha generata non puo' che essere il seguente. Il Basic ha interpretato la terna [3,5,60] come il comando di inserire alla linea 3 l'istruzione [,5,60], del tipo sconosciuto al suo repertorio; ma soprattutto le regole sintattiche non sono state rispettate. Infatti subito dopo il numero di linea deve sempre comparire o una parola chiave del Basic o il nome di una variabile scritta con certe convenzioni. Sul modo lecito di scrivere i nomi delle variabili diremo al paragrafo 2.6: qui basta ricordare che devono cominciare con una lettera dell'alfabeto. E a questo riguardo la nostra istruzione degenere conteneva proprio questo tipo di errore. Per venirci incontro e sollevarci da alcune manovre il Basic, rilevato l'errore di sintassi in quell'istruzione, immagina che vogliamo subito correggerla. A questo scopo predispone gia' l'ambiente EDIT che lo consente.

Per farla breve e rimettere le cose a posto e' sufficiente una semplice manovra. Per prima cosa occorre uscire dall'ambiente EDIT, perche' non abbiamo alcuna intenzione di correggere un'istruzione indesiderata: per far cio' digitiamo [CR], confermando in tal modo questa linea 3. Per curiosita', provate a listare il vostro programma con il comando LIST [CR]. Vi comparira':

```
 3 ,5,60
10 INPUT DV,DO,RP
20 CR=2700/RP
30 LG=10*CR/DO
40 HG=10*RP/DV
50 PRINT LG,HG
```

Avrete quindi la prova che questa famigerata linea 3 e' stata veramente inclusa nel programma. Per cancellarla sara' sufficiente, secondo quanto gia' esposto al paragrafo 2.3, battere [3] seguito da [CR]. Chiedete ancora di listare il programma per verificare che effettivamente la linea indesiderata sia stata cancellata. Ora potete tornare a usare il programma.

Vorremmo commentare anche le modalita' di visualizzazione che la PRINT consente, ma lo faremo al capitolo 3 dove tratteremo un'ampia casistica di impiego di questa fondamentale istruzione Basic. Mettiamo in pratica, invece, qualche altro comando.

Vorreste salvare il vostro programma registrandolo per sempre su un dischetto? Procuratevi un dischetto gia' pronto (si dice formattato) per la registrazione. Altrimenti avventuratevi al paragrafo 10.1, per ottenerlo a partire da un dischetto vergine. Introducete questo dischetto formattato nel drive a, inventate un nome da dare al programma -potremmo chiamarlo [gabbia]- e battete il comando:

```
SAVE"gabbia                [CR]
```

Vedrete accendersi la lampadina rossa del drive interessato (a): cio' significa che l'operazione di registrazione ha avuto seguito. Aspettate che si spenga. Verificate, per sicurezza, che nel dischetto sia stato veramente registrato un "file" con il nome [gabbia]. Per far cio' basta che digitiate il comando:

```
FILES                      [CR]
```

e se, sul video, tra le altre informazioni di servizio, rintraccerete il nome [gabbia], sarete sicuri che il vostro programma e' in salvo. Ora puo' anche mancare la corrente elettrica: il programma in memoria centrale sara' perso, ma non quello sul dischetto.

Come fare a richiamarlo? Per fare questa prova dato che il programma appena registrato e' ancora in memoria **cancelliamolo col comando [NEW]** seguito da [CR]. Verifichiamo che in memoria non ci sia piu' niente: adesso battendo [RUN] [CR], infatti, non otterrete altro che la risposta [Ok].

A questo punto, digitate il comando:

```
LOAD"gabbia                [CR]
```

E' una specie di comando duale del SAVE per quanto riguarda il senso del trasferimento dei dati tra memoria e dischetto; infatti, al contrario, di LOAD che carica in memoria, SAVE scarica su dischetto. Appena attivato un LOAD si accende subito la lampadina del drive interessato a significare, questa volta, un'operazione di lettura, anziche' di scrittura. Dopo pochissimi secondi il programma e' nuovamente in memoria centrale. Listatelo con un LIST [CR] e comparira' subito sul video. Se il richiamo dal dischetto aveva lo scopo di mandarlo subito in esecuzione, anziche' un comando LOAD, avreste potuto dare alla macchina il comando:

```
RUN"gabbia                 [CR]
```

che implicitamente esegue il caricamento da dischetto in memoria e quindi lo lancia in esecuzione. In tal caso avreste ricevuto subito il segnale [?] sul video.

Se poi volete avere la lista stampata del vostro programma sulla PR 15B, battete allora:

```
LLIST                        [CR]
```

Naturalmente la stampante deve essere collegata a M24, accesa e con un foglio inserito.

Se invece volete avere una **registrazione** stampata di tutto il vostro dialogo con M24, basta fare la manovra [CTRL] + [PrTSc], cioe' battere il tasto [PrtSc], mentre tenete premuto il tasto [CTRL].

Da quel momento la stampante e' asservita. Dovrete ovviamente tener d'occhio la lunghezza della carta, perche', appena manca, un sensore interno alla stampante "sente" l'anomalia e mettendo fuori servizio la stampante blocca anche il funzionamento di M24. Per rimettere le cose a posto ridate carta alla PR 15B e azionate il tasto AUTO-LOCAL sul lato anteriore della stampante, per spegnere la lampadina del blocco di file carta. Se poi volete momentaneamente sganciare la PR dall'M24 e' sufficiente ripetere la manovra [CTRL] + [PrtSc].

2.6 REGOLE ORTOGRAFICHE E D'INTERPUNZIONE

Le parole chiave e i nomi delle variabili di un programma possono essere scritti in lettere maiuscole o minuscole; in ogni caso vengono registrate in memoria come se fossero state tutte introdotte con lettere maiuscole.

Il nome della variabile **non** puo' essere **piu' lungo di 40 caratteri**, con alcune restrizioni:

- deve cominciare con una lettera alfabetica,

- nel corpo del nome, oltre alle lettere alfabetiche, sono ammesse solo le cifre numeriche e il punto,

- un eventuale carattere finale tra i seguenti

  ! % # $

  specifica la qualita' della variabile in merito agli impieghi cui e'destinata. Di cio' parleremo in altra sede.

Si noti che anche per gli spazi dobbiamo rispettare le regole sopraindicate.

Pertanto un nome di variabile che ha nel suo interno uno spazio non viene accettato.

Inoltre:

- almeno uno spazio deve precedere e seguire una parola chiave

- gli spazi all'interno dei numeri (compresi i numeri di linea) e delle parole non sono consentiti.

Un'altra possibilita' del Basic, infine, e' quella di inserire **in una qualunque linea** di programma un opportuno commento. In tal modo si puo' documentare un programma descrivendo le funzioni e gli effetti proprio in margine alle istruzioni che li provocano. Non c'e' alcuna restrizione di impiego di caratteri nell'ambito del campo commento: occorre soltanto definire l'inizio del campo con il carattere ['].

Per quanto riguarda l'ampiezza del campo esiste solo il limite dei

255 caratteri della linea di programma.

Ad esempio:

```
50 PRINT A 'stampa del valore attuale di A.
```

Dopo il campo commento, l'eventuale spazio residuo della linea di programma non puo' essere utilizzato da altre istruzioni; l'effetto di questi campi appare sul listing del programma, ed e' nullo sull'esecuzione del programma.

Ultimo rilievo importante: nello scegliere nomi per le variabili non si possono usare quelli delle parole chiave o dei comandi del Basic, a meno che non vengano fusi con altre parole.

Ad esempio: PRINTOR e SPRINT possono essere usate come nomi di variabile anche se la prima e l'ultima parte del nome coincidono con la parola chiave PRINT.

CAPITOLO
TRE

LE FRASI ELEMENTARI DEL BASIC SU M24

Ogni programma da scrivere e' come un esperimento su un banco di prova: l'istruzione e' l'oggetto del desiderio cognitivo. Di essa vorremmo saper ogni cosa. Come si muove al minimo e al massimo delle sue possibilita'. Ma per scoprire tutte le sue capacita' elaborative conviene provarla un po' per volta. Senza mettere tutti i suoi ingredienti al fuoco. Soprattutto per quelle istruzioni "tuttofare" che sembrano automobili di fantasia con dieci marce avanti e cinque retromarce. Conviene cominciare a impiegarle in modo ridotto, muovendo solo una "leva" alla volta, o al massimo due, con le relative combinazioni.

Solo cosi' potrete notare l'effetto particolare legato a una certa manovra.

Sarete, inoltre, maggiormente stimolati a questa indagine se, muovendo un parametro (variante dell'istruzione), l'effetto sara' diverso da quello previsto, tanto piu' se scoprirete qualcosa di spettacolare. In tal caso vi verra' in mente di utilizzarla in certe altre occasioni, per altri programmi...

3.1 I DIVERSI MODI DI INTRODURRE I DATI

L'introduzione dei dati per un elaboratore costituisce un'operazione fondamentale, potremmo dire vitale.

Poiche' e' l'uomo a fornire dati in diverse circostanze e in tempi diversi, iniziamo con una classificazione dei mezzi utilizzati.

Da questa analisi e dalle numerose applicazioni saranno chiari anche le modalita' e i momenti tipici in cui i dati dovranno essere introdotti. L'esecuzione di un programma puo' richiedere dati di diversa natura, e provenienti:

- da tastiera, per assegnare i valori alle variabili, o per dare i vari comandi necessari alle varie fasi di elaborazione;

- dallo stesso programma, sede di particolari informazioni, registrate al momento della scrittura;

- da dischetto, per caricare il programma, a seguito di un comando impostato da tastiera;

- ancora da dischetto, per trasferire un archivio di dati da elaborare durante l'esecuzione di un programma;

- da sistema operativo per attingere risorse macchina necessarie al programma.

Per l'emissione dei risultati sara' piu' opportuno parlare di supporti di destinazione, tra i quali:

- il video
- il dischetto
- il sistema operativo
- la stampante.

Per ciascuno dei casi sopra elencati esistono istruzioni o coppie di istruzioni preposte alle attivita' corrispondenti.

Nei prossimi capitoli tratteremo le varie situazioni nell'ambito di semplici programmi, in modo che il significato e le modalita' operative di ogni singola istruzione possano essere finalizzate verso un particolare effetto desiderato.

Tali programmi sono di tipo propedeutico e quindi non necessariamente ottimizzati (anche se possono diventarlo): se lo fossero gia' sarebbero talmente impenetrabili da non poter essere impiegati per uso didattico.

3.1.1 DATI DA TASTIERA

Abbiamo gia' usato l'istruzione INPUT nell'esempio introduttivo del paragrafo 2.5.

Riprendiamo tale esempio per apportarvi qualche modifica che, da un lato ne migliori l'impiego, e, dall'altro, dia la possibilita' di illustrare alcuni dei talenti residui di questa basilare istruzione.

Nella sua forma piu' elementare l'istruzione INPUT fa si' che M20 visualizzi [?], sospendendo ogni attivita' finche' l'operatore non abbia digitato il valore da introdurre nella variabile specificata.

In tale forma e' stata impiegata nell'esercizio "gabbia" del paragrafo 2.5 a cui nuovamente ci riferiamo.

```
10 INPUT DV,DO,RP
20 CR=2700/RP
30 LG=10*CR/DO
40 HG=10*RP/DV
50 PRINT LG,HG
Ok
RUN
? 3,5,60
 90           200
Ok
```

La prima osservazione da fare e' sul modo di chiedere i dati. E' ovvio che, a distanza di tempo, l'utilizzatore di un programma contenente la INPUT in questa forma possa non ricordare il significato delle variabili che stanno dentro il programma.

Altrettanto dicasi per il loro numero e l'ordine con cui sono state dichiarate. E non parliamo poi del tipo di separatori.

Meglio sarebbe se fosse lo stesso programma, nel momento in cui chiede i valori, a evidenziare sul video, nel modo piu' chiaro e colloquiale, a quale variabile si sta riferendo. E cosi' per tutte le altre variabili in gioco a cui e' necessario assegnare valori in ingresso.

Le possibilita' di arricchimento dell'istruzione INPUT sono tali che,

come abbiamo prima avvertito, non riteniamo conveniente esaurirle tutte. Ci limiteremo pertanto a illustrare solo alcuni sviluppi dell'istruzione che costituiranno poi uno standard di riferimento per tutti gli esercizi presentati in questo libro.

Vediamo quindi come si puo' rendere piu' leggibile (soprattutto per quell'utilizzatore che non sa come e' stato fabbricato) anche un semplice programma come quello finora sviluppato.

Nella figura 3.1, sono illustrate le variazioni inseribili nel corpo della stessa istruzione INPUT. Ad esempio possiamo scrivere:

```
INPUT "DENS.VERT.=";DV
```

"DENS.VERT." e' una frase scritta tra virgolette ["], notazione convenzionale che il Basic interpreta come estremita' per delimitare un campo di caratteri da stampare "pari pari" sul video prima del [?].

All'istruzione INPUT apparira' sul video il simbolo [?] per invitare l'operatore a introdurre il valore della variabile. Questa volta pero' sapremo -perche' e' lo stesso programma che ce lo ricorda- a quale tipo di variabile si riferisce il dato introdotto.

Ogni medaglia ha pero' il suo rovescio!
Infatti, in un'istruzione INPUT cosi' congegnata e' bene limitarsi a dichiarare non piu' di una variabile per volta, per non complicare troppo la scrittura della frase tra virgolette.

Allora la frase INPUT DV,DO,RP, per diventare piu' esplicita, verra' cosi' riscritta:

```
INPUT "DENS.VERT.=";DV
INPUT "DENS.ORIZZ.=";DO
INPUT "RIGHE/PAG."=";RP
```

Diamo un nuovo nome -e sia **gabbia1**- a questa prima versione migliorata del programma **gabbia** e inseriamolo come commento in una linea del programma, ad esempio la 5.

Per far cio' seguiamo quanto abbiamo indicato al paragrafo 2.6. In tal modo, dopo le opportune correzioni, il programma sara':

```
5 'gabbia: calcolo al variare delle densita' verticale
6 '        e orizzontale e del numero di righe per pa-
7 '        gina.
10 INPUT "DENS.VERT.";DV
20 INPUT "DENS.ORIZZ.";DO
30 INPUT "RIGHE/PAG.";RP
40 CR=2700/RP
50 LG=10*CR/DO
60 HG=10*RP/DV
70 PRINT HG
80 PRINT LG
Ok
RUN
DENS.VERT.? 3,5,60
?Redo from start
```

```
LOAD"gabbia1
Ok
LIST
10 'gabbia1: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 INPUT "DENS.VERT.";DV
50 INPUT "DENS.ORIZZ.";DO
60 INPUT "RIGHE/PAG.";RP
70 CR=2700/RP
80 LG=10*CR/DO
90 HG=10*RP/DV
100 PRINT HG
110 PRINT LG
Ok
RUN
DENS.VERT.? 3
DENS.ORIZZ.? 5
RIGHE/PAG.? 60
 200
 90
```

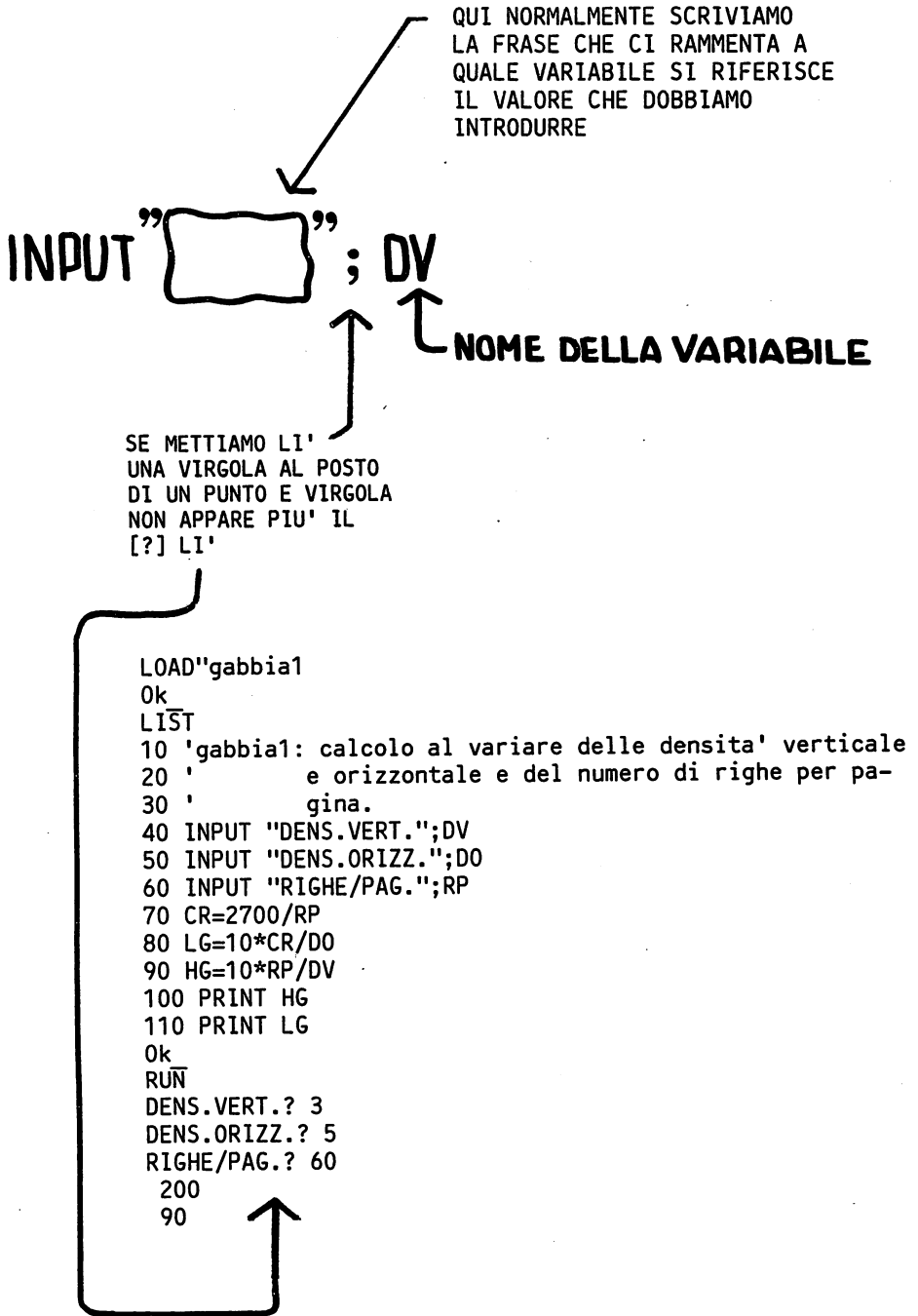



Figura 3.1

```
DENS.VERT.? 3
DENS.ORIZZ.? 5
RIGHE/PAG.? 60
 200
 90
Ok_
```

Come si puo' notare nella prima battuta, avendo risposto con piu' di un valore alla richiesta, l'elaboratore da' giustamente un messaggio di errore e un invito a ricominciare tutto daccapo.

Con il comando RENUM ordiniamo poi al Basic di rinumerare tutte le linee **di 10 in 10** a partire **da 10.** Ottengo cosi' lo stesso programma nella nuova sequenza:

```
10 'gabbia1: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 INPUT "DENS.VERT.";DV
50 INPUT "DENS.ORIZZ.";DO
60 INPUT "RIGHE/PAG.";RP
70 CR=2700/RP
80 LG=10*CR/DO
90 HG=10*RP/DV
100 PRINT HG
110 PRINT LG
Ok_
```

Facendo eseguire il programma, notiamo subito che ora, per ogni variabile assegnata, occorre concludere con [CR]. Potremmo inoltre decidere di sopprimere anche il [?]. La cosa e' del tutto lecita, visto che l'istruzione INPUT ha una variante che lo prevede. Infatti, come illustrato nella figura 3.1, prima del nome della variabile, e' sufficiente sostituire il separatore [;] con il separatore [,].

Nell'opera di miglioramento del dialogo tra la macchina e l'operatore, possiamo anche modificare la riga del video in cui far apparire la frase di richiamo. Ad esempio, se mettiamo un [;] subito dopo le parole chiave INPUT, le frasi di richiamo dell'INPUT successiva appariranno affiancate e tutte sulla stessa riga.

Cosi' facendo, pero', le frasi e i valori introdotti per le variabili risulterebbero troppo ravvicinati. Il rimedio a questo inconveniente e' un piccolo artificio che consiste nell'introdurre alcuni spazi di separazione in testa alle frasi di richiamo.

Le migliorie introdotte appaiono chiaramente nella figura 3.2.In essa notiamo un altro difetto che riguarda la stampa dei valori di HG sulla stessa riga dei valori di INPUT.

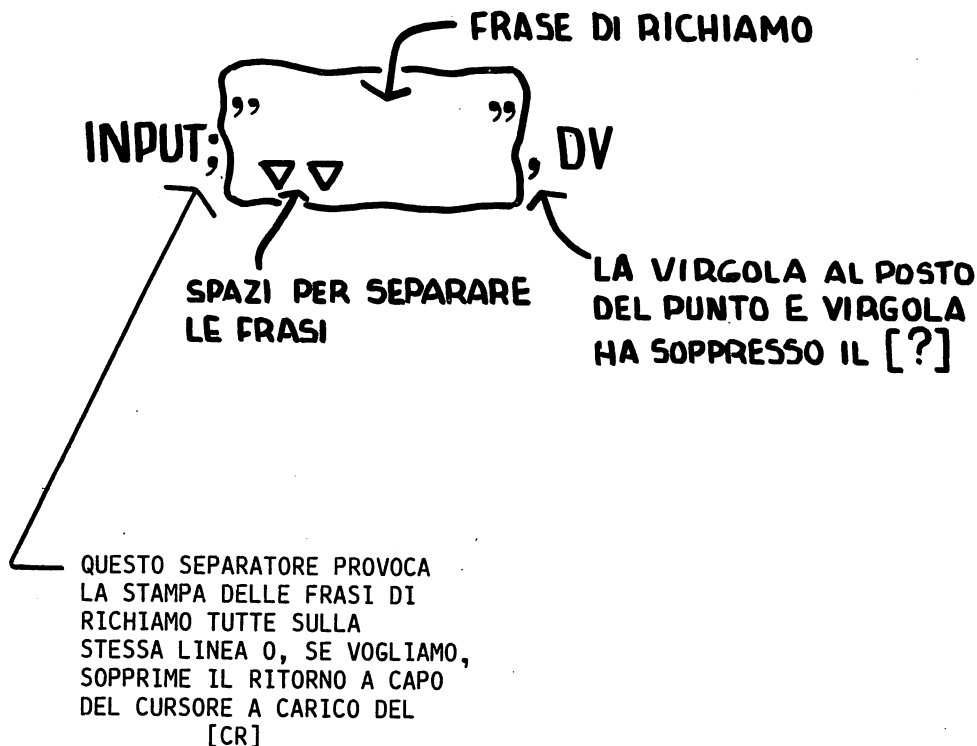
Per mettere le cose a posto e' sufficiente sopprimere il [;] di troppo nell'istruzione alla linea 60. Per far cio' listiamo il programma gabbia2 e una volta comparso sul video spostiamo il cursore in modo che lampeggi sotto il carattere [;] della linea 60 e cancelliamolo col tasto [DEL]. Convalidiamo la correzione col [CR] e saltiamo col cursore alla linea 10 per apportare anche li' una piccola correzione: si tratta di modificare nel commento il nome del

```
10 'gabbia2: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 INPUT;"DENS.VERT.=",DV
50 INPUT;"  DENS.ORIZZ.=",DO
60 INPUT;"  RIGHE/PAG.=",RP
70 CR=2700/RP
80 LG=10*CR/DO
90 HG=10*RP/DV
100 PRINT HG
110 PRINT LG
Ok
RUN
DENS.VERT.=3  DENS.ORIZZ.=5  RIGHE/PAG.=60 200
 90
Ok_
```

Figura 3.2

programma che, a causa della lieve modifica alla linea 60 e' diverso dal programma originario. Volendo rinominarlo gabbia2a (e come tale lo registreremo sul dischetto) per coerenza apporteremo tale nome anche nel commento. Per far cio', traguardato col cursore la posizione del carattere [:], entreremo in "modo inserzione" col tasto [INS] e poi battiamo [a]. Infine, convalidiamo anche questa correzione col [CR]; se poi vogliamo collaudare il programma mandandolo in esecuzione, bisogna, come al solito battere RUN seguito da [CR]. Tale manovra di lancio va fatta solo se il cursore lampeggia in una riga di di video completamente libera di caratteri.

Poiche' dopo la convalida della correzione alla linea 10 il cursore lampeggia nella prima posizione della linea 20 di gabbia2a (cioe' sotto il 2), si puo' cancellare il contenuto video della riga in almeno due modi:

1) battere il tasto [z] tenendo premuto [CTRL], e con questa manovra spariscono dal video (non dalla memoria!) tutto cio' che esiste nelle righe sottostanti quella in cui lampeggia il cursore;

2) battere il tasto [ESC], e con cio' si cancella sul video solo la riga in cui lampeggia il cursore.

Ora e' possibile lanciare il programma e, verificato che la correzione funziona, il programma puo' essere registrato col nuovo nome [gabbia2a] con la manovra:

```
Save"gabbia2a          [CR]
```

Gli esempi considerati fin qui dovrebbero avervi gia' dimostrato la grossa differenza che esiste tra un linguaggio naturale e uno formale. Aggiungiamo ora una piccola ma praticissima variante al programma, legata sostanzialmente al modo di impiegarlo. Per apportare la modifica, anticipiamo la conoscenza di un'altra istruzione Basic che fa parte del gruppo delle istruzioni di controllo che vedremo al paragrafo 5.3.

Si tratta dell'istruzione **GOTO** -fusione delle parole inglesi GO e TO- che significa: **andare a.** Alla parola chiave GOTO bisogna far seguire uno spazio e un numero di linea. Tale istruzione viene comunemente chiamata di **SALTO INCONDIZIONATO** perche' quando il Basic la incontra, nell'esecuzione di un programma, **salta** subito alla linea specificata dalla GOTO ignorando qualsiasi altra istruzione che precede tale numero di linea. Naturalmente la linea a cui saltare deve essere realmente esistente nel programma, altrimenti il Basic segnala l'errore: **UNDEFINED LINE NUMBER.**

Quali possono essere i motivi per impiegare una GOTO nel programma gabbia? Avrete certamente notato, quando volete introdurre una nuova terna di valori, la noia di dover digitare ogni volta il comando RUN. Avete anche visto che, se non lo fate, la terna puo' essere interpretata come una nuova istruzione che si aggiunge indebitamente al programma e che quindi deve essere cancellata.

Per non correre tutti questi rischi introduciamo allora una GOTO 40 alla linea 120, in coda al programma: quando il Basic la interpreta salta indietro all'istruzione 40, consentendoci di sperimentare una

nuova terna, senza il fastidio di rilanciare il programma con la RUN. Il ciclo continua indefinitamente perche' ogni volta tutte le variabili vengono riinizializzate, dandovi nuovi risultati da confrontare coi precedenti.

A questo punto se volete, potete interrompere il processo, senza spegnere la macchina e farlo in modo soffice con la solita manovra [CTRL C].

Terminiamo questo paragrafo dedicato all'istruzione INPUT per descrivere una sua funzione di utilita', attinente non tanto all'introduzione dei dati, quanto al suo impiego per la gestione di un programma. Infatti si puo' sfruttare la proprieta' sospensiva della INPUT per bloccare artificiosamente un programma in punti opportuni, anche se non abbiamo niente da introdurre. Per fabbricare l'artificio descritto basta scrivere in una linea del programma una INPUT cosi' fatta:

```
INPUT;"PREMERE [CR] PER CONTINUARE",A
```

dove A e' una variabile che non riguarda il programma e potremmo chiamarla variabile sospensiva. In effetti si crea comunque un deposito corrispondente in memoria; il programma quando incontra questa INPUT si ferma in attesa di ricevere qualcosa da passare al contenitore A.

Per sbloccare il programma e farlo riprendere non importa cercare che cosa introdurre: basta concludere premendo [CR]. In tal modo, anche senza introdurre niente, l'istruzione viene comunque soddisfatta, e il programma riprende a girare.

3.1.2 DATI DA PROGRAMMA

A prima vista si direbbe un'affermazione ovvia quella di poter considerare un programma come una sorgente di dati per l'elaboratore. Ma quanto qui vogliamo far rilevare riguarda un modo particolare di assegnare i dati di partenza al programma.

Le istruzioni INPUT, esaminate nel precedente paragrafo, permettono di passare dati ad un programma lanciato in esecuzione attraverso la tastiera: quando una INPUT viene incontrata nel programma, l'elaborazione viene sospesa e M24 attende che l'operatore introduca i dati richiesti. Ad ogni successiva esecuzione del programma e' possibile assegnare valori diversi alle variabili che compaiono nella INPUT.

Questa liberta' di scelta da parte dell'operatore e' una delle caratteristiche piu' interessanti dell'istruzione e dello stesso linguaggio Basic.

In questo modo di lavorare, tra l'altro, risiedono e si esaltano le proprieta' interattive della macchina.

La selezione dei valori segue l'indagine critica dei risultati che in sostanza guidano il processo stesso di risoluzione di un problema.

Questa possibilita' diventa meno interessante in quei problemi dove la maggior parte dei valori da assegnare a certe variabili sono gia' noti a priori o, per la natura dell'applicazione, debbono rimanere costanti nei vari esperimenti di calcolo.

In tal caso, per assegnare valori alle variabili, si usa una coppia di istruzioni Basic scritte nelle linee di un programma registrato in memoria. Si tratta delle istruzioni [READ] e [DATA] che agiscono nel seguente modo.

La [READ] assegna alle variabili i valori che legge dalle istruzioni [DATA]. La macchina, in questo particolare processo di acquisizione dati, si comporta in modo analogo a un carillon, in cui i pioli del rullo sono le variabili e le lamelle i valori da assegnare (vedi figura 3.3).

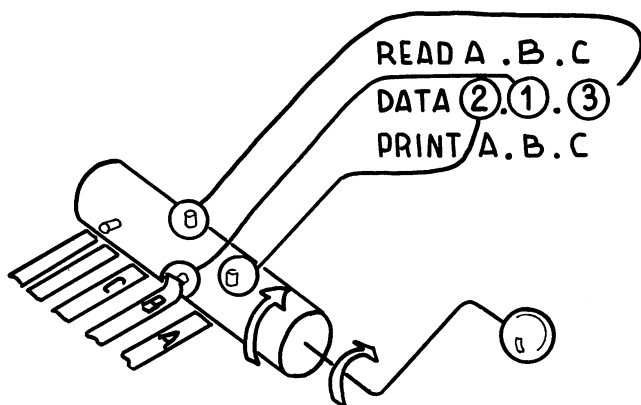



Figura 3.3

Nel seguente programma **volt** l'istruzione READ ha facolta' di accedere a una o piu' istruzioni DATA, che saranno lette nell'ordine in cui sono scritte. Se vi sono piu' variabili che dati, il Basic segnalera' l'errore con un messaggio "OUT OF DATA": se, invece, vi fossero meno variabili che valori, dopo l'esecuzione della READ, i valori non assegnati verranno trascurati.

```
10 'volt:esempio di READ...DATA
20 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 110,125,160,220
Ok_
RUN
 110           125           160           220
Ok_
```

Come potete vedere la sintassi da rispettare e' molto semplice. Nella READ si elencano i nomi delle variabili separati dalle virgole, e i valori sono letti nella DATA alla linea 40, nell'ordine in cui si desidera che vengano assegnati.

La PRINT alla linea 30 consente la visualizzazione dei valori delle variabili. Avrete anche notato che le istruzioni PRINT sono state scritte prima delle DATA. Tuttavia il programma ha ugualmente prodotto risultati: cio' significa che l'interprete Basic prima di eseguire il programma prende in carico i valori di eventuali DATA da assegnare alle READ.

Tali valori costituiscono una successione il cui ordine e' quello in cui sono stati scritti nelle DATA.

Se in un programma esistono piu' istruzioni DATA, l'ordine dei valori e' quello con il quale vengono letti, linea per linea, fino alla fine del programma.

La successione dei valori DATA, come prima dicevamo, viene costruita appena il programma viene lanciato con il [RUN] [CR] ed e' a disposizione di qualunque READ la voglia leggere.

Per rendere un po' piu' flessibile l'utilizzo delle istruzioni READ...DATA esiste una terza istruzione, che agisce di conserva con le prime due.

Si tratta dell'istruzione RESTORE. Con questa istruzione si indica alla macchina qual e' il primo elemento da trattare tra quelli contenuti in una DATA. Cosi' la READ, piazzata dopo una RESTORE torna a leggere il primo elemento della successione dei valori DATA, anche se sono stati assegnati in altre READ precedenti. Vediamo, ad esempio, dopo l'introduzione di una RESTORE, come funziona il programma [volt]:

```
10 'volt1:esempio di READ...DATA...RESTORE
20 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 110,125,160,220
50 RESTORE 70
60 READ A,B,D
70 DATA 115,140,175,380
80 PRINT A,B,D
90 DATA 107,127
100 DATA 149,179
Ok
RUN
 110            125            160            220
 115            140            175
Ok
```

In questo nuovo programma, denominato **volt1**, la RESTORE specifica addirittura la linea in cui e' scritta la prima DATA che la READ deve leggere per assegnare i valori alle sue variabili. Al lancio del programma vengono stampate due righe di risultati: la prima si riferisce a valori stampati dalla PRINT alla 30, assegnati alle variabili elencate nella READ, che, a sua volta, li ha letti nell'ordine specificato alla linea 40. La seconda riga dei risultati, invece, e' l'effetto della PRINT alla linea 80 dove i valori delle variabili A,B,D sono stati letti dalla READ alla linea 60, secondo quanto stabilito dalla RESTORE. La READ infatti, legge le istruzioni DATA a partire dalla linea 70, consegnando i valori alla PRINT che li stampa assegnandoli alle variabili A,B,C.
Per questo i risultati stampati sono 115,140,125 e non di nuovo 110,125,160. Se vogliamo che il programma faccia proprio questo, basta cancellare il numero 70 nella RESTORE.

La cancellazione dell'intera linea 50 provoca, invece, l'effetto di trascurare tutti i valori contenuti nelle DATA scritte prima della nuova READ (nel programma in esame, la READ alla linea 60). I seguenti programmi **volt2** e **volt3** contengono entrambe le varianti.

```
10 'volt2:esempio di READ...DATA...RESTORE
20 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 110,125,160,220
50 RESTORE
60 READ A,B,D
70 DATA 115,140,175,380
80 PRINT A,B,D
90 DATA 107,127
100 DATA 149,179
Ok
RUN
 110           125           160           220
 110           125           160
Ok
```

```
10 'volt3:esempio di READ...DATA...
20 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 110,125,160,220
60 READ A,B,D
70 DATA 115,140,175,380
80 PRINT A,B,D
90 DATA 107,127
100 DATA 149,179
Ok
RUN
 110           125           160           220
 115           140           175
Ok_
```

Un ultimo rilievo: la READ alla linea 60 chiede al Basic di leggere i primi tre valori nella successione delle DATA, indipendentemente dalla storia dei valori che a quelle variabili sono stati associati dalla precedente READ scritta alla linea 20. Infatti, questa READ alla linea 60 **vede** la successione delle proprie variabili A,B,D: i valori da associare sono quelli che si trovano nella successione delle DATA gestita dalla RESTORE. A maggior chiarimento degli effetti che la RESTORE provoca sulle READ, vediamo ancora i seguenti programmi:

```
10 'drr2:ancora test data/read/restore*********
20 DATA 7,-5,12,2,6
30 READ B,A,C
40 RESTORE
50 READ D
60 PRINT A;B;D
61 PRINT D;B;A
Ok
RUN
-5  7  7
 7  7 -5
Ok_
```

```
10 'revolt:esempio di READ...DATA...RESTORE
20 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 110,125,160,220
50 RESTORE 70
60 READ A,B,D
70 DATA 115,140,175,380
80 PRINT A,B,D
90 DATA 107,127
100 DATA 149,179
110 RESTORE
120 READ P,Q,R,S,T,U,V,W,X,Y,Z
125 PRINT
130 PRINT P Q R S T U V W X Y Z
Ok_
RUN
 110           125           160           220
 115           140           175

 110  125  160  220  115  140  175  380  107  127  149
Ok_
```

Quest'ultimo programma differisce dai precedenti per il ripristino alla linea 50 dell'istruzione RESTORE 70 e l'aggiunta, qua e la', di altre linee DATA, nonche' di una RESTORE e una READ di ben 11 variabili in coda. In quest'esempio, alla linea 110 appare chiaramente il ruolo di segnaposto dell'istruzione RESTORE. Poiche' la RESTORE e' generica e non specifica in quale particolare linea deve piazzarlo, implicitamente la macchina posiziona tale segnaposto sul primo elemento della prima linea che contiene un'istruzione DATA. Questo e' il motivo per cui alla prima variabile designata P che compare nella READ della linea 120 viene assegnato il valore 110. Altrettanto dicasi per le altre 10 variabili che vengono stampate a carico della PRINT della linea 130. Degli impieghi pratici di questa terna di istruzioni tratteremo per esteso piu' avanti.

3.1.3 DATI DA DISCHETTO

Nella microinformatica, il dischetto e' un importante, anzi insostituibile, supporto ausiliario. Il suo utilizzo e' multiforme e adattabile a vari impieghi. Ripiloghiamo, adesso, i momenti in cui finora abbiamo fatto lavorare i dischetti.

- All'accensione della macchina, il bootstrap loader ha caricato il nucleo di MS-DOS dal dischetto in cui tale sistema operativo e' registrato.

- Durante lo svolgimento di certe attivita', sempre dal dischetto di MS-DOS, abbiamo caricato in memoria quei componenti del sistema normalmente non facenti parte del nucleo (ad esempio il gwbasic).

- Per salvare un programma appena costruito in memoria o per richiamarlo in tempi successivi, abbiamo usato un dischetto, normalmente distinto da quello del sistema.

Al paragrafo 2.4 sono gia' state, in parte, descritte le piu' significative attivita' di scambio dati con i dischetti; ma quelle piu' importanti - in quanto legate alle specifiche applicazioni- vengono svolte per la registrazione di dati direttamente strutturati dall'utente, come gli archivi clienti, fornitori, fatture etc. Su tali argomenti e' dedicato l'intero capitolo 8, a cui rimandiamo per tutti i dettagli.

Tornando alle funzioni del sistema operativo, ricordiamo che oltre alle informazioni sulle risorse software, molte attivita' riguardano anche gli stessi componenti hardware della macchina e il loro modo di operare.

Ad esempio, in qualunque momento e' possibile ottenere -attraverso speciali moduli di MS-DOS normalmente residenti solo sul dischetto sistema- modificazioni che influenzano il funzionamento logico e fisico della macchina, come:

- caratteristiche grafiche associate alla stampante collegata (comando **mode**);

- capacita' a riprodurre per punti su stampante grafica il contenuto del video (comando **graphics**);

- caratteristiche del Basic (comando **gwbasic**).

3.2 I DIVERSI MODI DI ESTRARRE LE INFORMAZIONI

L'edificio elaboratore ha uscite principali e secondarie in relazione agli scopi per i quali e' stato programmato. Ad esempio i consuntivi delle spese di esercizio condominiale devono chiaramente essere stampati per la consegna ai singoli proprietari; al contrario, le sperimentazioni per modelli in certe applicazioni di fisica possono essere seguite meglio attraverso rapide escursioni grafiche sul video e eventualmente stampate per hard-copy nelle situazioni di arrivo di una serie di prove.

Oltre alle uscite visibili attraverso le unita' periferiche classiche, esistono quelle molto piu' elementari avvertibili solo per l'accensione della spia luminosa del driver interessato alla segnalazione del flusso dei dati provenienti dalla memoria centrale di M24.

Le uscite verso il sistema operativo, parte delle quali sono state brevemente riportate al termine del paragrafo 3.1.3, riguardano gli assetti di macchina hardware e software e particolari manovre per le riassegnazioni dei tasti funzionali (F1÷F10) e la generazione di catene di comandi MS-DOS mediante le quali e' possibile fabbricare dei supercomandi personali (batch files). Questi ultimi, veri ambienti operativi, costruiti a misura di utente, possono notevolmente ridurre le fasi di attrezzaggio del vostro laboratorio di informatica M24 e darvi, ogni volta che accendete la macchina, tutte

le risorse non standard che abitualmente impiegate, senza necessita' di richiamarle singolarmente dal dischetto MS-DOS con le relative manovre.

Di tali operazioni tratteremo nel capitolo 10, dedicato ai "segreti di bottega" e nell'Appendice B dedicata alle manovre MSDOS.

Nei successivi paragrafi, tratteremo dell'istruzione PRINT e derivate che, insieme con la INPUT, sono le fondamenta del linguaggio Basic.

3.2.1 USCITA SU VIDEO

Per visualizzare dati sullo schermo di M24 esiste in gwbasic l'istruzione PRINT, che avete gia' utilizzato per estrarre i risultati nell'esempio trattato al paragrafo 2.5. L'istruzione PRINT puo' essere abbreviata col simbolo [?]; anche in questa forma l'abbiamo gia' usata al paragrafo 1.4.1 per i calcoli immediati. Ora e' tempo di impiegarla nelle sue espressioni piu' complete.

Innanzitutto diciamo che da sola puo' visualizzare una lista di tanti elementi quanti ne puo' contenere una linea di programma. Ad esempio l'istruzione

```
PRINT A,B,C,D,E,F
```

interessa sei variabili numeriche, ma potrebbe contenerne molte di piu' e non solo variabili. Ad esempio

```
PRINT A;2;125;B
```

produce la visualizzazione di

```
 0  2  125  0
```

Si noti che il valore 0 (zero) attribuito alle variabili A e B si spiega col fatto che, in mancanza di valori loro assegnati, l'unico che possano avere e' appunto lo zero corrispondente a "contenitore vuoto". Sono da notare nella PRINT:

a) i separatori degli elementi della lista, in particolar modo gli ultimi, che influenzano la posizione in cui vengono stampati;

b) la possibilita' di mettere al posto di uno o piu' elementi espressioni, anche complesse, da calcolare;

c) l'uso combinato con la Input per particolari effetti tipografici.

L'uso del punto e virgola come separatore fa si' che gli elementi vengano presentati ravvicinati sul video.
Siccome i numeri possono essere positivi o negativi, i primi saranno

preceduti da uno spazio, mentre i secondi avranno un segno meno.
Se viene usata la virgola anziche' il punto e virgola, per separare gli elementi, questi appariranno meno vicini tra loro e saranno distanziati di 14 posizioni. Se sono in numero tale da non essere contenuti nella stessa riga, andranno a invadere righe successive del video.
Facciamo qualche esempio fuori programma, usando cioe' le istruzioni in modo diretto, immediatamente eseguibili.

Esempio 1

```
print 125;7*7;2^4;49-2+16;                                [CR]
 125   49   16   63
Ok
```

Nella Print vengono dichiarate e invitate a comparire sul video espressioni costanti come 125 oppure risultati di calcoli su numeri.
Inviando la Print cosi' scritta al Basic, con il solito [CR], otterremo l'effetto sulla riga del video sottostante a quella in cui abbiamo impostato la Print.

Esempio 2

```
a=2^4:b=7^9:c=(54-5)^(-1/3)                               [CR]
Ok

print a;b;c;a;15*15                                       [CR]
 16   4.035361E+07   .2732759    16   225
Ok
```

In questo esempio abbiamo fatto precedere la Print da tre istruzioni separate dai due punti [:], per la creazione in memoria di tre variabili **a**, **b** e **c**, alle quali abbiamo assegnato valori pari al risultato di operazioni sui numeri che compaiono al secondo membro.
Nella Print che compare subito dopo abbiamo quindi chiesto di "vedere" il contenuto delle variabili **a**, **b**, **c**, ancora **a** e, infine, il risultato di **15x15**.
Come potete vedere i contenuti di **b** e **c** sono espressi in semplice precisione con notazione esponenziale (vedi capitoli 1 e 6). Noterete anche gli spazi tra un numero e l'altro per accogliere l'eventuale segno meno.

Esempio 3

```
Print  a,b,c,a,2                                          [CR]
 16     4.035361E+07      .2732759       16      2
```

Questo esempio differisce dal precedente soprattutto per aver sostituito il punto e virgola con la virgola, come separatori tra gli elementi della Print. In tal modo lo spazio tra gli elementi sara' maggiore e pari a 14 caratteri.
Gli esempi successivi descrivono con maggior dettaglio, le regole sintattiche della Print.

```
c1=(54-5)^(-1/2):due.alla .decima=2^10:tre.al.cubo=3^3  [CR]
Ok

due.alla.18=2^18
Ok

Print a;b;c;c1; due.alla.decima;tre.al.cubo;due.alla.18
16   4.035361E+07    .2732759     .1428572   1024   27   262144
Ok
```

I nomi delle variabili come [due.alla.decima] sono stati ovviamente scelti non per comunicare alla macchina l'operazione da svolgere -che e' invece indicata in modo non ambiguo al secondo membro in notazioni Basic- ma per dimostrare quanto possa essere utile scegliere un nome di variabile sufficientemente descrittivo.

Le seguenti ultime tre Print, infine, mostrano esempi di calcolo di espressioni.

Nell'ultima un rilievo particolare: contiene la variabile a il cui contenuto e' ancora quello assegnato poco prima con la relazione

```
a=2  4=16.


print (54-5)^1/2
 24.5


print (54-5)^(1/2)
 7
Ok


print ((54-5)^(1/2))^(a-14)
 49
Ok
```

Dopo aver descritto i contenuti logici della Print esaminiamo ora alcune delle varianti grammaticali previste che insieme con l'uso di particolari separatori producono effetti speciali nella presentazione dei risultati.

Supponiamo di voler visualizzare il contenuto di una certa variabile A. Abbiamo visto che e' sufficiente dichiarare il nome della variabile nel corpo dell'istruzione Print.

Cio' nonostante, nella sua forma piu' elementare, l'istruzione visualizza solo il valore numerico della variabile, senza aggiungere altro.

L'operatore, in queste condizioni, legge dei numeri sul video e deve ricordare a quali variabili appartengono.

Anche nell'esempio semplice della gabbia con due soli valori per risultato (HG e LG) bisogna ricordarsi quale viene stampato per primo. In caso di dubbio, basta leggere il programma con comando List per chiarire il particolare.

C'e' comunque un metodo piu' semplice per facilitare la lettura dei risultati: lo prevede la stessa Print in una delle sue numerose varianti sul tema.

Tale metodo prevede l'impiego nel corpo della Print di una descrizione libera immediatamente prima del nome della variabile. Tale descrizione viene racchiusa tra speciali separaratori (le virgolette ["]), di cui abbiamo gia' parlato per analoghi motivi quando abbiamo trattato l'istruzione Input. L'effetto delle virgolette e' di consentire l'emissione fedele sul video di quanto esse racchiudono (vedi figura 3.4).

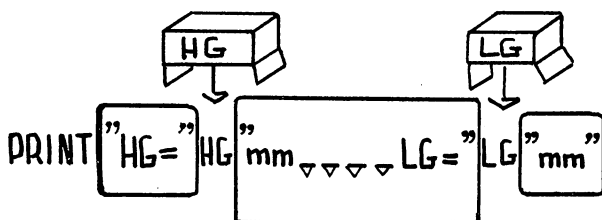
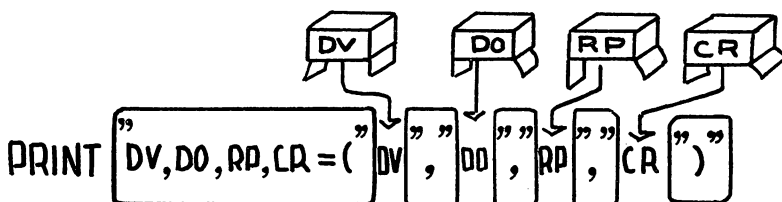



Figura 3.4

In tal modo, una Print "A="A provochera' la visualizzazione del contenuto di A preceduto dai caratteri compresi tra le virgolette. Se A vale, ad esempio 200, verra' visualizzato

```
A=200
```

Per evitare eccessive complicazioni mnemoniche, non intendiamo trattare tutte le possibili varianti della Print, molte delle quali potrebbero adesso risultare superflue.

Seguono i programmi **gabbia4** e **gabbia5** come esempi di utilizzo della PRINT; in particolare, in gabbia5, sono state aggiunte le linee 120 e 130 per mostrare come la PRINT possa migliorare la presentazione dei risultati.

Si osservi, inoltre, come la PRINT alla linea 115 serva a creare una riga vuota nel risultato.

```
LOAD"b:gabbia4
Ok_
LIST
10 'gabbia4: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 INPUT;"DENS.VERT.=",DV
50 INPUT;"  DENS.ORIZZ.=",DO
60 INPUT"  RIGHE/PAG.=",RP
70 CR=2700/RP
80 LG=10*CR/DO
90 HG=10*RP/DV
100 PRINT "HG="HG
110 PRINT "LG="LG
120 PRINT "HG="HG"  LG="LG
Ok_
RUN
DENS.VERT.=2  DENS.ORIZZ.=5  RIGHE/PAG.=25
HG= 125
LG= 216
HG= 125   LG= 216
Ok_
```

```
LOAD"b:gabbia5
Ok_
LIST
10 'gabbia5: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 INPUT;"DENS.VERT.=",DV
50 INPUT;"  DENS.ORIZZ.=",DO
60 INPUT"  RIGHE/PAG.=",RP
70 CR=2700/RP
80 LG=10*CR/DO
90 HG=10*RP/DV
100 PRINT "HG="HG
110 PRINT "LG="LG
115 PRINT
120 PRINT "HG="HG"mm      LG="LG"mm"
130 PRINT "DV,DO,RP,CR=("DV","DO","RP","CR")"
Ok_
RUN
DENS.VERT.=2  DENS.ORIZZ.=5  RIGHE/PAG.=25
HG= 125
LG= 216

HG= 125 mm      LG= 216 mm
DV,DO,RP,CR=( 2 , 5 , 25 , 108 )
Ok_
```

3.2.2 USCITA SU STAMPANTE

Questo tipo di uscita, oltre alla disponibilita' fisica di una stampante funzionante (collegata a M24, accesa, con carta inserita), richiede alcune particolari predisposizioni o manovre per la sua pratica utilizzazione.

Se la stampante collegata e' la PR 15B si possono scegliere due modi di funzionamento: permanente o temporaneo. Per ottenere il primo tipo di funzionamento occorre battere il tasto PrtSc tenendo premuto il tasto CTRL: da quel momento qualsiasi attivita' di tastiera o di M24 che interessi il video (salvo la grafica) ottiene risposta oltre che su video anche su stampante. Per disattivare la stampante, che' altrimenti resterebbe sempre agganciata, basta rifare la stessa manovra con la quale era stata asservita.

Un ultimo modo di far lavorare la stampante e' quello di copiare fedelmente, punto per punto, il contenuto del video. In gergo informatico -lo ripetiamo- questa operazione equivale a fare l'hard-copy del video, una copia materiale, in contrapposizione con l'immagine su video, che, per sua natura, e' volatile: cioe', scompare se si spegne la macchina.

Per quanto riguarda i disegni, la manovra da fare e' battere ancora il tasto PrtSc, ma, questa volta, tenendo premuto il tasto SHIFT.
L'**hard-copy** della grafica avviene correttamente se, all'atto del caricamento del sistema operativo, e' stato richiamato il modulo GRAPHICS.

CAPITOLO
QUATTRO

L'AMBIENTE GRAFICO

Nei precedenti capitoli abbiamo illustrato i "mattoni" piu' elementari e piu' noti del linguaggio Basic: le istruzioni Input e Print. Introduciamo ora alcune nozioni fondamentali sulla grafica Basic M24: tale conoscenza rendera' possibile anche una piu' ricca metodologia di apprendimento.

L'approccio parte da una constatazione banale: avvalersi non solo delle forme tradizionali per valutare i risultati ma anche di mezzi piu' espressivi. La grafica gioca un ruolo importante nei processi logici e mnemonici. Con carta e matita s'insegna meglio che con la sola enunciazione verbale. Non importa se la rappresentazione grafica e' imperfetta: basta che colga l'essenza di un risultato. Ad esempio: disegnare l'equazione di una retta che passa per due punti del piano cartesiano.

Potreste non riuscire a trasformare un concetto matematico in forma grafica e non e' detto che sia sempre possibile o semplice. Ma se vi riuscirete non sarete solo capaci di utilizzare un mezzo di divulgazione scientifica piu' efficiente, ma -cosa piu' importante- avrete scavato fino alle radici del concetto. Fino a ieri si affermava: "Se non sei riuscito a spiegarti, vuol dire che neppure tu hai capito fino in fondo!" Oggi potremmo dire, con altrettanta sicurezza: "Se non sei riuscito a cavarci fuori neppure un grafico...", oppure: "Se il programma non gira, non e' colpa della macchina...."

Questo tentativo di fusione tra matematica e geometria non serve solo a proporvi una sorta di matematica costruttiva: serve anche a costringervi a far pratica dell'ambiente grafico di M24.

Il capitolo introduce gli elementi basilari di tale ambiente, nonche' alcune delle principali istruzioni grafiche di cui l'interprete **gwbasic** di M24 ampiamente dispone.

Non verranno trattati di tali istruzioni tutti i numerosi attributi grafici che ne estendono le possibilita' pittoriche, in quanto verranno ripresi al capitolo 8, dedicato appunto al colore.

4.1 IL PIXEL

Il pixel e' l'elemento grafico piu' semplice rappresentabile sul video di M24. Il termine deriva da due parole inglesi (**pict**ure **el**ement). Come abbiamo gia' detto al paragrafo 1.2.3, nella massima densita' grafica, che corrisponde anche alla massima capacita' di caratteri rappresentabili (25 righe di 80 caratteri) e' possibile accenderne fino a 256000. La possibilita' grafica di accendere uno qualsiasi di questi pixel si ha, tuttavia, solo se viene attivato il **modo grafico**. Esiste al riguardo il comando **screen** z, dove z puo' assumere i valori 0, 1, 2, 3. Per impartire a gwbasic l'ordine di lavorare in ambiente grafico con la massima risoluzione grafica (640x400 pixel) occorre dichiarare fuori programma o dentro un programma: **screen 3**.

In assenza di questa parola magica la combinazione che la macchina offre automaticamente senza predisposizioni particolari e' la screen

0, cioe' solo un modo **testo** di impiegare il video. Quando invece vogliamo selezionare la piu' bassa densita' grafica (320x200=64000 pixel), e contemporaneamente scegliamo anche la densita' del testo video corrispondente alla capacita' di 1000 caratteri (25 righe di 40 caratteri), e' sufficiente scrivere (sempre in ambiente gwbasic) il comando **screen 1.** Esiste, infine, la possibilita' di lavorare in media risoluzione (640x200=128000 pixel con testi di 25 righe da 80 caratteri) e il relativo comando e' **screen 2.**
Accendere un pixel sul video e' come fissare un punto su un piano cartesiano. Occorre naturalmente definire l'origine degli assi e il loro senso di orientamento.

Con riferimento alla figura 4.1, le convenzioni implicitamente assegnate da gwbasic sono il vertice in alto a sinistra del video per l'origine degli assi, da destra a sinistra il senso positivo dell'asse delle ascisse, e verso il basso quello dell'asse delle ordinate. Per inciso, tale scelta poco cartesiana e ha forse giustificazioni divulgative per necessita' di uniformarsi al criterio editronico europeo di contare le righe dall'alto verso il basso e i caratteri di una riga da sinistra verso destra. Cio' nonostante chiamaremo **ascissa** ancora la distanza di un punto dall'asse Y delle ordinate, e **ordinata** quella dall'asse X delle ascisse. I due nomi, ascissa e ordinata, si chiamano **coordinate** e mediante esse viene localizzato un punto nel piano cartesiano.

Sul video di M24 l'ascissa dei pixel visualizzabili puo' variare secondo il grado di risoluzione attivo in quel momento: avendo attivato la massima col comando screen 3, l'ascissa puo' variare da 0 a 639 e l'ordinata da 0 a 399. Con la scrittura P=(x,y) intendiamo definire un punto del piano a distanza x (ascissa) dall'asse verticale e a distanza y (ordinata) dall'asse orizzontale.

Ad esempio nella figura 4.1, le coordinate (99,199) sul video di M24 rappresentano un punto P che ha la distanza 100 pixel dall'asse verticale (delle ordinate) e distanza 200 pixel dall'asse orizzontale (delle ascisse). L'istruzione Basic per accendere un pixel e' trattata al paragrafo 4.4.1.

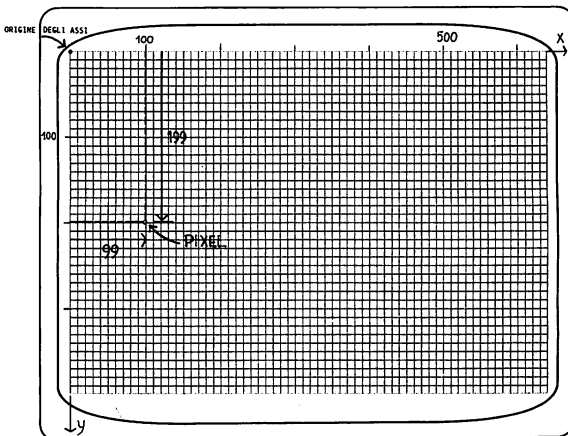



Figura 4.1

4.2 IL CURSORE

Il cursore e' un segnale grafico che indica qual e' la posizione sul video in cui apparira' il carattere che stiamo per digitare. Quando si vuole far apparire un dato sul video in una determinata posizione e' quindi necessario, prima dell'istruzione Print, definire la posizione del cursore, altrimenti M24, appena riceve il dato attraverso il buffer di tastiera, lo allinea sulla prima riga utile e a partire dalla prima posizione. Per fissare una particolare posizione del cursore sul video occorre ovviamente avere un sistema di riferimento. **Come per i pixel**, l'origine degli assi cartesiani per la localizzazione del cursore e' il vertice del video in alto a sinistra: l'asse delle y, che scandisce le righe, e' quindi ancora orientato verso il basso, cosi' come quello dei caratteri sulla riga e' ancora orientato positivamente verso destra.

Facciamo un esempio: supponiamo di essere in modo grafico a bassa risoluzione (25 righe di 40 caratteri ciascuna) e di voler stampare un [3] nella posizione 60 della decima riga. Per far cio', prima di una Print "3" dobbiamo posizionare il cursore esattamente all'ascissa 60, ordinata 10 (vedi figura 4.2).

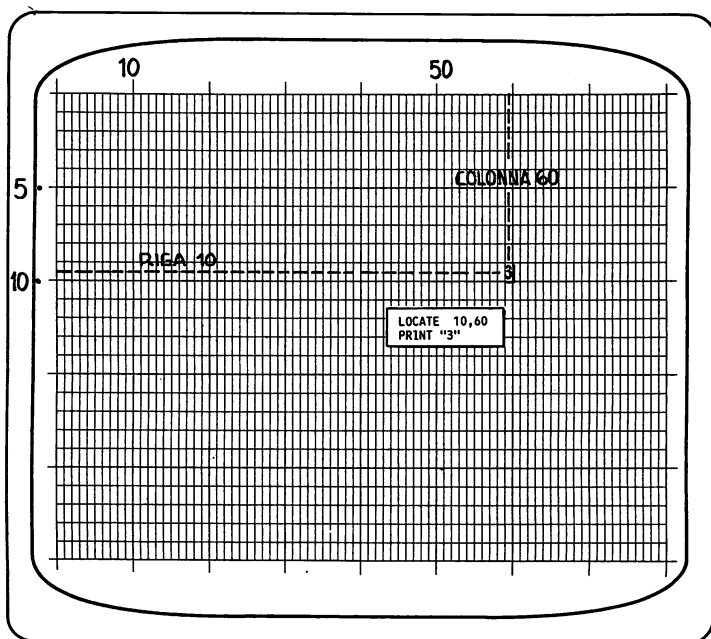



Figura 4.2

L'istruzione Basic disponibile per questa manovra e' la

```
LOCATE R,C
```

dove (R,C) sono le coordinate del cursore espresse in Riga e in Colonna. Le istruzioni da scrivere sono:

```
LOCATE 10,60:PRINT "3"   [CR]
```

Con l'istruzione CLS si azzerano i condizionamenti operati sul cursore che si riporta alla sua posizione di riposo, cioe' nella colonna uno della prima riga. Si noti che le coordinate per il posizionamento del cursore sono indipendenti da quelle per la localizzazione del pixel: le prime servono per la topologia dei caratteri di un testo, le seconde per quella di un disegno.

Si noti ancora che, al contrario dei pixel, la numerazione delle righe e delle colonne parte da 1 e non da zero.

Per far un po' di pratica su quanto abbiamo detto finora, facciamo girare il seguente programma denominato **curvar**.

```
10 'curvar:scrivere col cursore variab.
20       'a bassa risoluzione
30 CLS:KEY OFF:SCREEN 1
40 INPUT "riga (1...25)-> ",R
50 INPUT"colonna (1...40)-> ",C
60 INPUT "numero=",A
70 LOCATE R,C:PRINT "(numero)="A
80 INPUT "",W
90 LOCATE  10,35 :PRINT "2 x (numero)="2*A
100 INPUT"",W
110 LOCATE  18,10 : PRINT "1/(numero)="1/A
120 INPUT"",W
130 GOTO 30
```

```
riga (1...25)-> 9
colonna (1...40)-> 4
numero=123126299


   (numero)= 1.231263E+08

2 x (numero)= 2.462526E+08



         1/(numero)= 8.121741E-09
_
```

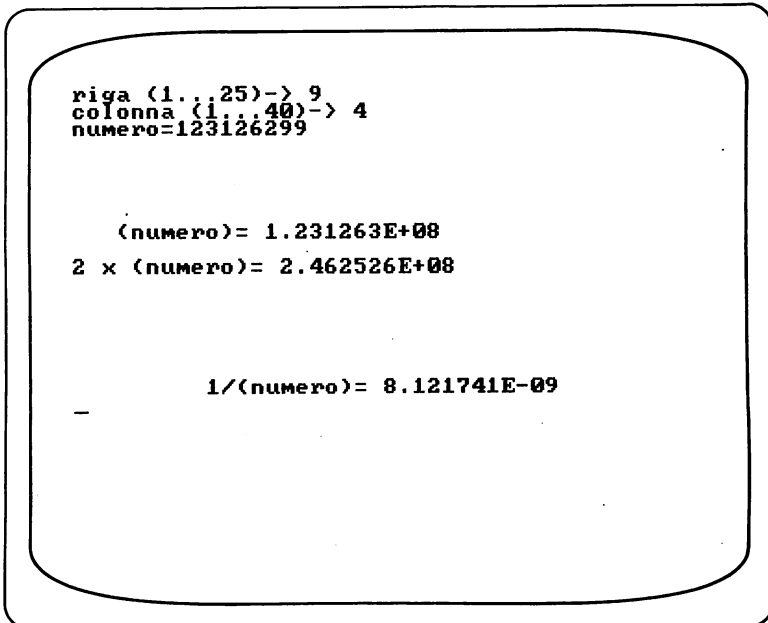



Figura 4.3

Un colpo d'occhio alle istruzioni e noterete che alla linea 70 abbiamo parametrizzato le variabili R e C, agganciandole ai valori introdotti con le corrispondenti Input alle linee 40 e 50. Con la Input della linea 60 introduciamo un numero che verra' associato alla variabile A. Lo stesso numero verra' stampato a carico della Print della linea 70. Le altre locate alle linee 90 e 110 sono invece bloccate e non sono parametrizzate ne' governabili dall'esterno del programma. Le relative Print, piazzate subito dopo, compiono alcune operazioni sulla variabile A. Alle linee 80, 100 e 120 le Input sospensive, con le variabili "rifiuto" W, per consentire di seguire, uno alla volta, l'effetto delle tre LOCATE (figura 4.3). Per non ingombrare inutilmente il video, il messaggio contenuto tra le virgolette di tali sospensive e' stato ridotto a niente. Le doppie virgolette, per motivi sintattici, non possono essere soppresse.

Una variante del programma appena descritto e' **curvar1**. In esso si puo' scegliere interattivamente il grado di risoluzione desiderata: alla linea 30 il valore 0, 1, 2 o 3 viene assegnato alla variabile z.

```
10 'curvar1:scrivere col cursore variab.
20 CLS:KEY OFF
30 INPUT"Risoluz.(0,1,2,3)= ",Z
50 SCREEN Z
60 IF Z>1 THEN K=80
65 IF Z<2 THEN K=40
70 INPUT "riga (1...25 )-> ",R
75 IF R=0 THEN 70
80 IF R>25 THEN 70
90 PRINT "col.(1..."K")-> ";
100 INPUT "",C
102 IF C=0 THEN 90
105 IF C>K THEN 90
110 INPUT "numero=",A
120 LOCATE R,C:PRINT "(numero)="A
130 INPUT "",W
140 LOCATE  12,11 :PRINT "2 x (numero)="2*A
150 INPUT"",W
160 LOCATE  20,11 : PRINT "1/(numero)="1/A
170 INPUT"",W
180 GOTO 20
```

Si noti nella 90 la variabile k il cui valore dipende dalla scelta di z e dalle conseguenti assegnazioni alle linee 60 e 65; rispetto a curvar sono state introdotte alcune protezioni sugli input di R e C per evitare messaggi di errore nel caso di scelta di valori non ammessi (la colonna 0 e quelle superiori alla massima ammessa per la risoluzione scelta, la riga 0 e quelle superiori alla 25).

A parita' di risoluzione si puo' scegliere una larghezza di riga utile minore di quella messa a disposizione. Nel programma curvar2 vediamo al lavoro questa funzione (linea 58) che viene attivata dal comando **width**: l'argomento puo' essere multiplo, ma nel caso qui trattato ci siamo limitati all'effetto pilotato da un solo parametro che viene scelto a carico dell'istruzione di Input alla linea 52.

Anche qui abbiamo inserito alcune protezioni per evitare gli errori di introduzione.

```
10 'curvar2:    scrivere col cursore variab.
20 CLS:KEY OFF' su video a larghezza variab.
30 INPUT"Risoluz.(1,2,3)= ",Z
40 IF Z=0 THEN  30
50 SCREEN Z
52 INPUT "Lar.=",W
53 IF Z><2 THEN 55
54 IF W=4 THEN 58
55 IF W<8 THEN 52
58 WIDTH W
60 IF Z><1 THEN K=80
65 IF Z=1 THEN K=40
70 INPUT "rig.(1... 25 )-> ",R
75 IF R=0 THEN 70
80 IF R>25 THEN 70
90 PRINT "col.(1..."K")-> ";
100 INPUT "",C
102 IF C=0 THEN 90
105 IF C>K THEN 90
110 INPUT "numero=",A
120 LOCATE R,C:PRINT "(numero)="A
130 INPUT "",W
140 LOCATE  12,11 :PRINT "2 x (numero)="2*A
150 INPUT"",W
160 LOCATE  20,11 : PRINT "1/(numero)="1/A
170 INPUT"",W
180 GOTO 20
```

Oltre a portare il cursore in una posizione desiderata puo' essere utile vincolare l'inizio di una successiva scrittura alla posizione raggiunta dalla precedente. Supponete, ad esempio introducendo un testo, di voler andare a capo dopo i due punti, e di voler riprendere tre righe dopo, incolonnando il testo tre posizioni a sinistra dei due punti. Per risolvere questo tipo di problemi editoriali occorre un aggancio tra due **locate.** E' cio' che realizzano le istruzioni CSRLIN e POS nelle espressioni seguenti:

```
RI=CSRLIN
CO=POS(0)
```

dove RI e CO assumono rispettivamente i valori della posizione attuale del cursore espressi in numero di riga e di colonna. Nel seguente programma **scrivar**, che nella parte iniziale e' identico al curvar, alla linea 90 appaiono appunto le due funzioni CSRLIN e POS che assegnano i valori alle due variabili RI e CO. Tali valori vengono anche visualizzati in posizioni dettate dalle locate alle linee 95 e 96.

La LOCATE della linea 110, che governa la posizione da cui iniziera' a operare l'Input del "secondo numero", e' agganciata alla scrittura precedente attraverso le funzioni CSRLIN e POS. In tal modo, una scrittura e' legata alla lunghezza o alla posizione raggiunta dalla precedente.

```
10 'scrivar:scrivere in bassa risoluzione col cursore
20 CLS      ' agganciato ai valori di POS e CSRLIN
25 KEY OFF:SCREEN 1
30 INPUT "riga (1...25)-> ",R
40 IF R=0 GOTO 30
45 IF R>25 THEN 20
50 INPUT"colonna (1...40)-> ",C
60 IF C=0 GOTO 50
70 INPUT;"primo numero= ",N1
80 LOCATE  R,C :PRINT "primo numero="N1;
85 '
90 RI=CSRLIN:CO=POS(0):PRINT
95 LOCATE 1,30:PRINT "rig.="RI
96 LOCATE 2,30:PRINT "col.="CO
100 INPUT "",W
105 IF CO-32<1 THEN CO=43
106 IF RI-5<1 THEN RI=10
110 LOCATE  RI-5,CO-32:INPUT;" secondo numero= ",N2
115 PRINT "="N2
120 INPUT"",W
130 GOTO 20
```

Il successivo programma scrivar1 e' la variante che consente di sperimentare l'ambiente in cui operano le istruzioni Basic che controllano il cursore ai vari livelli di risoluzione dello schermo.

```
10 'scrivar1:scrivere col cursore agganciato
20 CLS      ' ai valori di POS e CSRLIN
25 KEY OFF
30 INPUT "Risoluz..(0,1,2,3)= ",Z
50 SCREEN Z
60 IF Z>1 THEN K=80
61 IF Z<2 THEN K=40
62 INPUT "riga (1...25)-> ",R
63 IF R=0 GOTO 62
64 IF R>25 GOTO 62
66 PRINT  "col.(1..."K")-> ";
67 INPUT "",C
68 IF C=0 GOTO 66
69 IF C>K THEN 66
70 INPUT "primo numero= ",N1
80 LOCATE  R,C :PRINT "primo numero="N1;
90 RI=CSRLIN:CO=POS(0)
95 LOCATE 1,30:PRINT "rig.="RI
96 LOCATE 2,30:PRINT "col.="CO
100 INPUT "",W
105 IF CO+2>K THEN CO=1
106 IF RI+5>25 THEN RI=10
110 LOCATE  RI+5,CO+2:INPUT;"secondo numero= ",N2
115 PRINT "="N2
120 INPUT"",W
130 GOTO 20
```

Una passata del programma in screen 2 e' riportata in hard-copy nella figura 4.4.

Figura 4.4

Figura 4.4a

```
Risoluz.(1,2,3)= 1
riga (1...25 )-> 8
col.(1... 40 )-> 3
numero=123457899


  (numero)= 1.234579E+08


           2 x (numero)= 2.469158E+08




           1/(numero)= 8.099928E-09
```

Figura 4.4(b)

```
rig.(1... 25
)-> 8
col.(1... 80
)-> 3
numero=123457
899

  (numero)=
 1.234579E+08


2 x (numero)=
 2.469158E+08




1/(numero)=
 8.099928E-09
```

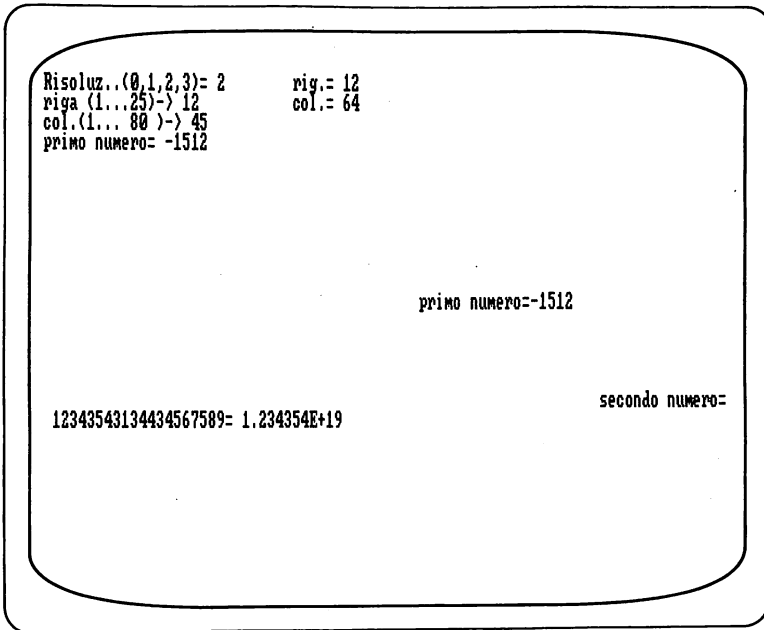
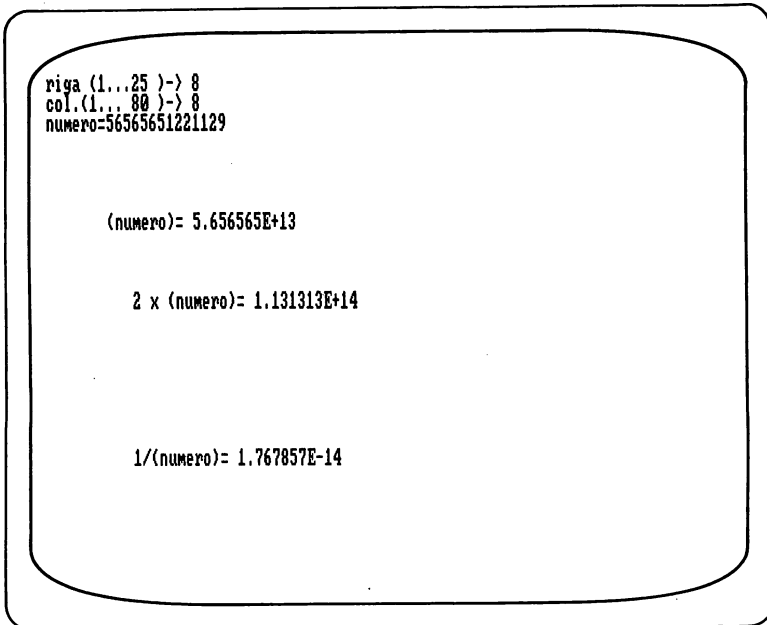
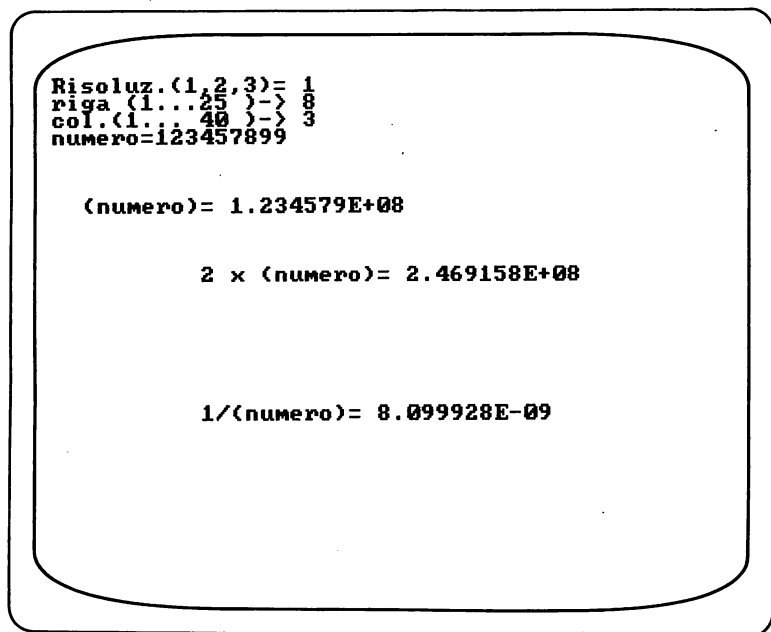
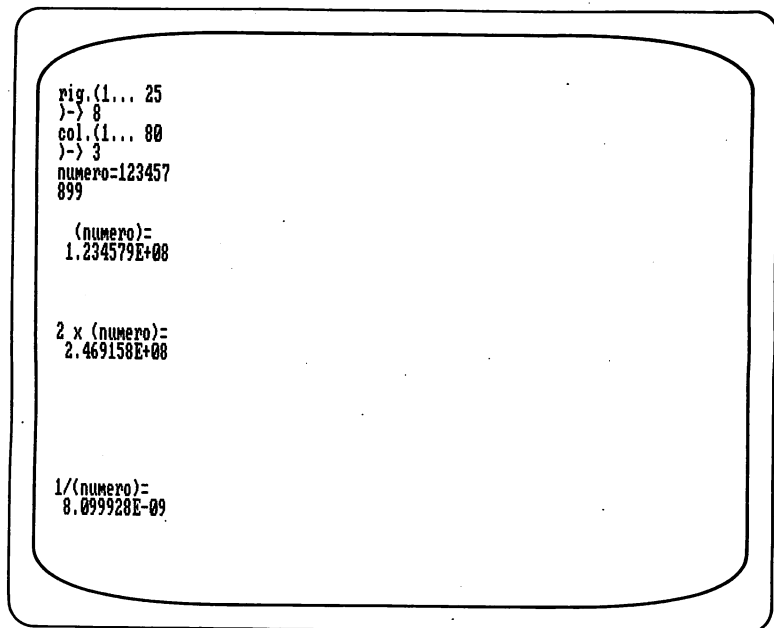



Figura 4.4(c)

4.3 LE FINESTRE

Il concetto di finestra su M24 da' enormi possibilita' di progettazione grafica: e' infatti possibile suddividere il video in un certo numero di parti o finestre. La suddivisione dipende dalla scelta fatta a inizio programma cosi' come la risoluzione grafica che vale per tutte le finestre aperte nello schermo fisico. La finestra di partenza e' ovviamente lo schermo fisico e ad esso, insieme con la risoluzione assegnata dall'istruzione screen, bisogna far riferimento per il dimensionamento delle altre.

Innanzitutto definiamo un po' di terminologia e alcuni criteri geometrici. Il primo criterio e' che M24 puo' disegnare una figura in almeno due modi: assumendo come spazio utile tutto lo schermo fisico, oppure solo una parte. Non solo, ma nel caso vogliamo che disegni in una porzione o finestra dello schermo fisico, e' possibile fare in modo che venga disegnata solo quella parte della figura che sta nella finestra e tutto il resto vada perduto, oppure si puo' ottenere la proiezione del disegno. Nel primo caso si ha un tassello della figura, nel secondo una sua riduzione in scala. La scelta di questi due modi di operare viene fatta, in ambiente gwbasic, con l'istruzione **view**, che ha la seguente sintassi:

```
view screen (v1,w1)-(v2,w2)
```

dove v1 e v2 sono le ascisse minima e massima dei punti della finestra, cioe' le distanze dei lati sinistro e destro della finestra dal bordo sinistro dello schermo fisico, e w1 e w2 le ordinate minima e massima dei punti della finestra, cioe' le distanze dei lati alto e basso della finestra misurate verticalmente a partire dal bordo superiore dello schermo fisico (figura 4.5).

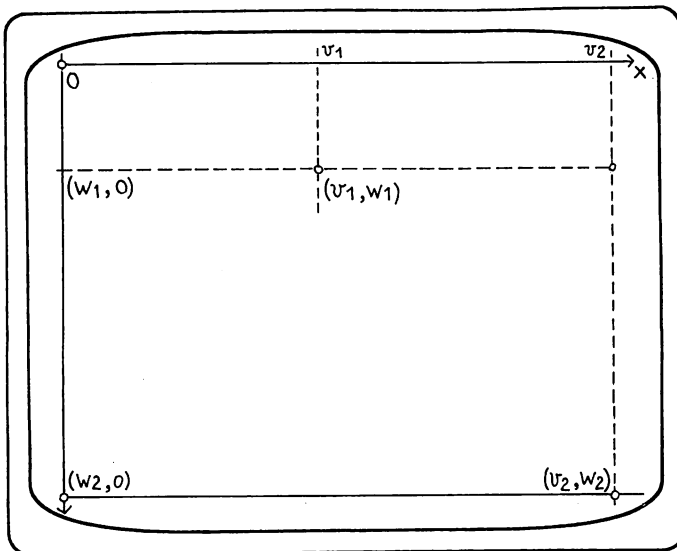



Figura 4.5

L'assenza della parola screen nell'istruzione view viene interpretata da M24 come ordine di disegnare in modo proiettivo, cioe' lo spazio di riferimento non e' piu' l'intero video ma solo quello definito dalle coppie di valori (v,w). Chiameremo, per semplicita', proiettiva tale finestra, per distinguerla da quella che e' solo un tassello dell'intero video. Si deve fare attenzione che, proprio perche' e' un tassello, quando non vi vediamo alcun risultato disegnato da M24, cio' puo' dipendere dal fatto che proprio in quella porzione dello schermo il programma grafico non ne prevede. Nella finestra proiettiva, invece, si otterra' la proiezione di cio' che si sarebbe ottenuto in uno schermo fisico di pari capacita'. Ovviamente, i limiti massimi naturali della finestra aperta con l'istruzione view sono quelli dell'intero schermo fisico. In tal caso, le coordinate dei vertici del contorno della finestra saranno quelli dello schermo fisico; piu' precisamente, anche se inutilmente, l'istruzione view dovrebbe scriversi:

```
view (0,0)-(639,399)
```

In figura 4.6 uno schema del funzionamento e dell'interpretazione geometrica dell'istruzione **view**.

Il vertice della piramide e' il punto di vista della figura rappresentata nel piano μ (che supponiamo quello dello schermo fisico di M24), mentre α e' quello della finestra proiettiva. Dalla figura si vede che la rappresentazione del triangolo riprodotto nel tassello appartenente al piano μ e' solo una parte della figura, mentre quella in α e' una sua proiezione.

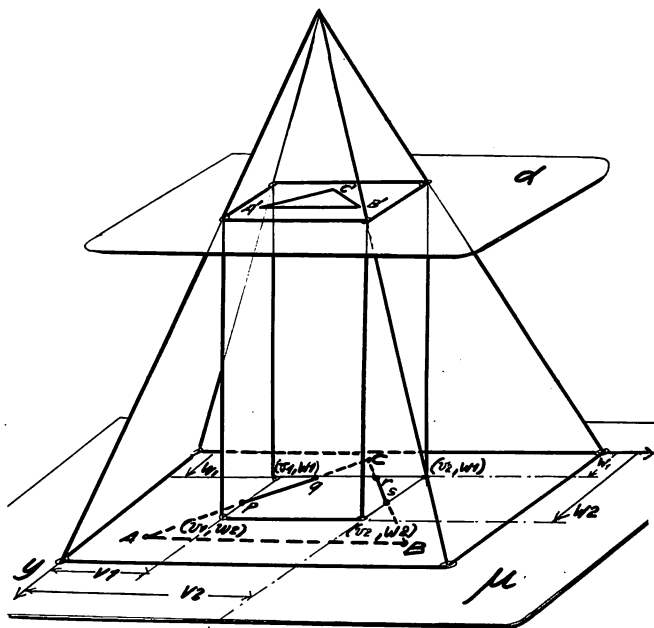



Figura 4.6

Quanto detto finora e' soltanto servito a ordinare a M24 di aprire finestre proiettive o non. Le istruzioni view corrispondenti andranno poste come enunciati in testa al programma o al sottoprogramma nel quale compaiono le vere elaborazioni grafiche. In altre parole, con la view abbiamo preparato il foglio di carta e il pantografo: per vederne gli effetti occorre attualizzarla nell'ambito di un programma nel momento in cui serve per ottenere un certo scenario.

Una volta attualizzata una certa finestra con la corrispondente istruzione view, dobbiamo ricordare che fino a quando non ne attiviamo una nuova, tutto verra' sempre riferito all'unica in essere. Per saltare a un'altra porzione del video bisogna definire un'altra finestra con la relativa **view**, tenendo conto che i bordi delle varie finestre possono sovrapporsi e che il programma vede una sola finestra attiva, nel momento in cui opera graficamente. Per meglio regolarsi sui limiti fisici delle singole finestre aperte, e' possibile chiedere all'istruzione view di disegnarne i bordi. Tale desiderio viene soddisfatto dichiarando la view nel seguente modo:

```
view (v1,w1)-v2,w2),,1
```

Tra le due virgole si puo' specificare il numero del colore con cui riempire la finestra.

Il passaggio da una finestra all'altra deve essere regolato da opportune istruzioni di salto finestra. La cosa notevole di questa scenografia e' che i contenuti delle altre eventuali finestre, precedentemente attivate, restano come congelati nelle sezioni corrispondenti del video.

Per mettere in pratica quanto abbiamo detto sulle finestre, esaminiamo il programma gabset. In tale programma alla linea 90 appare l'istruzione grafica piu' elementare: si tratta dell'istruzione PSET il cui argomento contiene le coordinate del pixel che si desidera accendere e che vengono chieste dal programma mediante le istruzioni Input alle linee 50 e 60.

```
10 'gabset: analisi delle finestre
20 KEY OFF:CLS
30 SCREEN 1
40 WIDTH 19
50 INPUT "ascissa=",X
60 INPUT "ordinata=",Y
70 VIEW (153,40)-(273,125),,1
80 'cls
90 PSET(X,Y)
100 PRINT
110 INPUT "(CR per continuare)",A
120 PRINT
130 GOTO 50
```

Come si puo' vedere dall'hard-copy di una passata del programma (figura 4.7) l'effetto grafico avviene dentro la finestra proiettiva e i pixel vengono accesi secondo un riferimento dato dalla nuova origine degli assi che e' localizzata alle coordinate fisiche (153,40). Rispetto a tale nuova origine il vertice opposto ha nuove coordinate (120,85).

Tali coordinate si ottengono calcolando le misure effettive, espresse in pixel, dei lati della finestra (273-153=120; 125-40=85).

Si noti che la WIDTH alla linea 40 limita a 19 i caratteri delle righe dedicate all'introduzione dei dati: in tal modo, raggiunta la linea 25, lo scorrimento verticale del quadro interessa solo quella fascia laterale dello schermo che non disturba l'area destinata alla grafica. Alla linea 80 l'istruzione CLS, se fosse attivata, cancellerebbe solo il contenuto della finestra.

Nell'hard-copy di figura 4.7 sono state introdotte le coordinate dei quattro punti vicini agli spigoli della finestra: cio' dimostra che se l'ascissa e/o l'ordinata di un punto superano i valori delle coordinate del vertice (120,85), il punto non potra' essere rappresentato nella finestra.

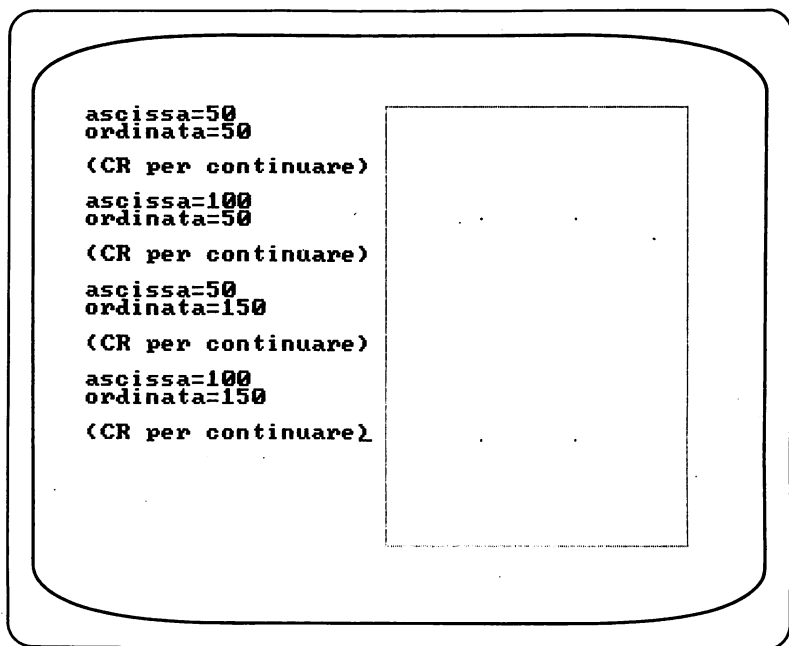



Figura 4.7

Nella figura 4.8 riportiamo una rappresentazione proiettiva della finestra definita dalla view in un programma.

153
273
0
319
40
(153,40)
85
125
(120,85
120
153,40
85
273-153=120
150,85
125 - 40
40
125
153
273
318

Figura 4.8

Riprendiamo ora il programma **gabbia**, che ridenominiamo **gabset3** e modifichiamolo in modo che i risultati si formino in una certa finestra individuata dalle coordinate di due suoi vertici opposti che dichiariamo nell'istruzione **view** piazzata alla linea 150.

In tale finestra in particolare vogliamo visualizzare i risultati estratti dalle Print alle linee 120 e 130. Lo scenario va progettato con le istruzioni finora conosciute.

Facciamo ancora pratica di PSET col solito programma gabbia nel quale possiamo includere le seguenti tre nuove linee

```
170 PSET(100,20)
180 PSET(100+LG/2,20)
190 PSET(100,20+HG/2)
```

Con la PSET alla linea 170 (vedi la figura 4.9) localizziamo il punto alto a sinistra della gabbia di coordinate (100,20), che costituira' il punto origine di riferimento per il punto alto a destra (PSET alla linea 180) e il punto piu' basso a sinistra (PSET alla linea 190).

Dopo queste modifiche il programma, ridenominato **gabset3**, si presenta nel modo seguente.

```
10 'gabset3: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 KEY OFF:CLS
50 SCREEN 1:WIDTH 19
60 INPUT "DENS.VERT.";DV
70 INPUT "DENS.ORIZZ.";DO
80 INPUT "RIGHE/PAG.";RP
90 CR=2700/RP
100 LG=10*CR/DO
110 HG=10*RP/DV
120 PRINT HG
130 PRINT LG
140 'WINDOW (0,0)-(639,399)
150 VIEW (160,1)-(318,198),,1
160 CLS
170 PSET (100,20)
180 PSET (100+LG/2,20)
190 PSET (100,20+HG/2)
200 PRINT
210 INPUT "(CR per continuare)",A
220 PRINT:GOTO 60
```

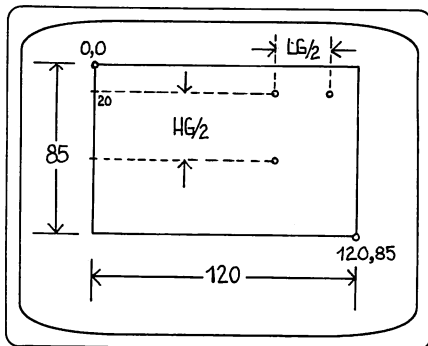



Figura 4.9

Si noti che per maggior chiarezza avremmo potuto scrivere:

```
170 PSET(100,20)
175 X=LG/2
180 PSET(100+X,20)
185 Y=HG/2
190 PSET(100,20+Y)
```

In tal forma e' piu' immediato accorgersi che il punto acceso dalla PSET della linea 180 non e' il punto gia' acceso dalla 170, ma uno con ascissa che dipende dal valore di X e quindi per la 175 da LG.
Analogo ragionamento puo' ripetersi per la PSET alla linea 190, dove il punto acceso, rispetto ancora all'origine stabilita dalla PSET(120,20), ha un'ordinata che dipende dal valore di Y e cioe' dal valore di HG per le relazioni contenute alla 185. Nel programma **gabset3** abbiamo invece omesso le istruzioni di assegnazione alle linee 175 e 185 e abbiamo direttamente sostituito nelle PSET le variabili HG e LG. Poiche' HG e LG sono le stesse variabili del problema **gabbia**, ad ogni esecuzione del programma vorremmo che i risultati contenuti in HG e LG andassero ad accendere dei pixel sul video: la distanza dei due pixel dall'origine stabilita nel punto (100,20) darebbe una risposta grafica immediata al problema e una sua diretta interpretazione allo spazio utile dedicato alla stampa e a quello della pagina che lo ospita. Il modo di impiegare le istruzioni grafiche appena definite nel programma **gabbia** di partenza deve essere pero' legato al momento in cui esse entrano in scena durante l'esecuzione del programma. Vediamo dove. La scelta va fatta pensando all'effetto che si vuole raggiungere, come sempre quando si scrive un programma. Un ragionevole impiego delle due finestre puo' essere il seguente. Come in gabset, la finestra dedicata all'introduzione dei dati si ottiene con la width alla linea 50; l'altra per i risultati attraverso l'istruzione view alla linea 150. Il risultato grafico, contenente anche il dialogo di alcune passate, appare in hard-copy nella figura 4.10. Come si puo' notare, ogni volta che si vuole sperimentare un'altra combinazione di parametri, basta, come invita la linea 210, premere il tasto [CR] e si torna alla linea 60.
Arrivati alla linea 160 si cancella nella finestra grafica il contenuto del precedente esperimento e cosi' via.

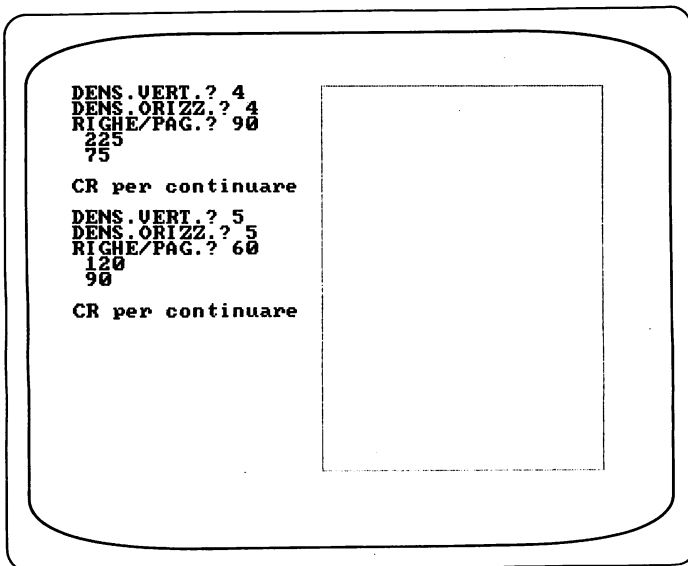



Figura 4.10

Una variante notevole, sul piano della gestione grafica, di gabset3 e' quella realizzata con **gabset4.**

```
10 'gabset4: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina.
40 KEY OFF:CLS
50 SCREEN 1:WIDTH 19
55 VIEW PRINT 10 TO 25
60 INPUT "DENS.VERT.";DV
70 INPUT "DENS.ORIZZ.";DO
80 INPUT "RIGHE/PAG.";RP
90 CR=2700/RP
100 LG=10*CR/DO
110 HG=10*RP/DV
120 PRINT HG
130 PRINT LG
140 WINDOW (0,0)-(639,399)
150 VIEW (160,1)-(318,198),,1
152 PSET (100,20)
154 PSET (100+LG/2,20)
156 PSET (100,20+HG/2)
158 INPUT "",A
160 CLS
162 VIEW (1,10)-(95,50),,1
163 PSET (100,20)
164 PSET (100+LG/2,20)
165 PSET (100,20+HG/2)
168 VIEW (100,10)-(150,50),,1
170 PSET (100,20)
180 PSET (100+LG/2,20)
190 PSET (100,20+HG/2)
200 PRINT
210 INPUT "(CR per continuare)",A
220 PRINT:GOTO 60
```

Le novita' sono:

- l'istruzione view print che limita la finestra di introduzione dei dati al gruppo di linee nelle quali il testo deve scorrere. Nel caso specifica di gabset4 dalla linea 10 all 25.

- alla linea 140 e' stata attivata l'istruzione window che ha la stessa sintassi della view con la quale convive per dare la completa disponibilita' proiettiva alla finestra la cui cornice e' definita dalla view stessa.
  In particolare la window (0,0)-(639,399) concede alla view successiva di rappresentare una figura aggiustando la scala in larghezza e in altezza in relazione ai lati della finestra dichiarata. Dei cambiamenti di scala, comunque, tratteremo il dettaglio al paragrafo 5.4.

- l'introduzione di altre due finestre con le corrispondenti view alle linee 162 e 168; a ciascuna di esse segue lo stesso pacchetto di PSET che consente di riprodurre gli stessi risultati grafici nell'ambito dello spazio fisico assegnato alle singole finestre.

In figura 4.11 l'hard-copy di alcune passate di gabset4.

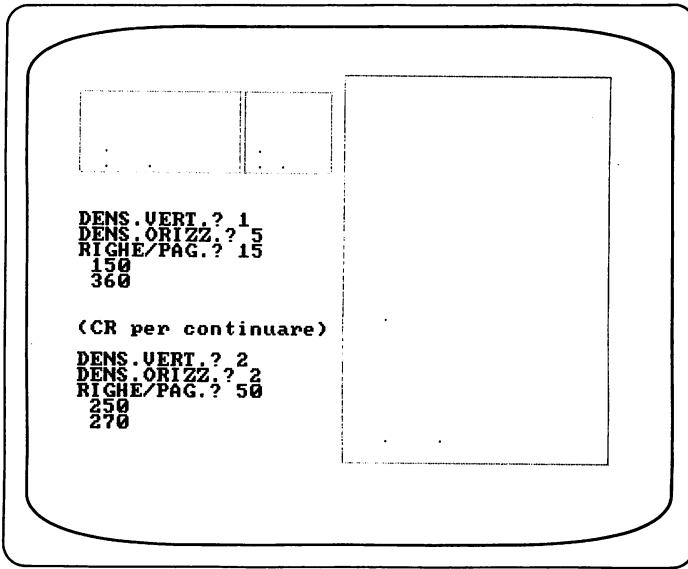



Figura 4.11

Sul funzionamento della nuova istruzione window e' opportuno fare alcune considerazioni preliminari per poi trattarne piu' dettagliatamente in seguito in tema di cambiamenti di scala.

Innanzitutto notiamo che quando in un programma viene usata la window tutte le istruzioni grafiche scritte nelle linee successive a quella in cui si trova la window disegnano simmetricamente rispetto all'asse delle ascisse (orizzontale), comportandosi, quindi diversamente dal loro normale funzionamento.

In altre parole l'asse delle ordinate riacquista la convenzione cartesiana di essere orientato verso l'alto, anziche' verso il basso e l'origine degli assi e' localizzato nel vertice inferiore sinistro dello schemo di M24, anziche' in quello superiore.

La presenza dell'istruzione window in un programma puo' comunque produrre **effetto scala** anche se non e' seguita dalla view: in tali condizioni lo schermo di riferimento su cui effettuare il cambiamento di scala avverra' non piu' sulla finestra, ma sull'intero schermo fisico.

4.4 LE FRASI GRAFICHE

Da questo momento proviamo a disegnare sul video facendo uso delle altre istruzioni grafiche che insieme alla PSET costituiscono il bagaglio minimo del repertorio grafico di M24. Cominceremo quindi con le piu' semplici frasi della grafica contenute nel linguaggio GWBASIC M24, che sono: PSET, PRESET, LINE e CIRCLE.

Della prima si e' gia' fatto un breve cenno in chiave finestre, ma ne riprendiamo ora sistematicamente la trattazione.

Con la PSET e la PRESET, rispettivamente, si accende o si spegne un pixel sul video, con la **line** si possono fare rette e rettangoli, e con la **circle** circonferenze e ellissi. In ciascuna di esse sono contenute varianti che ne ampliano le possibilita' grafiche e con le loro combinazioni si puo' praticamente disegnare qualunque cosa.

Nei successivi paragrafi le useremo gradualmente con la nota tecnica del mattone sopra mattone.

4.4.1 IL PUNTO

Abbiamo gia' definito il punto grafico o pixel. L'istruzione grafica per governarlo e' la PSET (abbreviazione di point set= posizionamento del punto). Il formato e' il seguente:

```
PSET(x,y)
```

in cui x e y sono le coordinate del punto del video (espresse in unita' compatibili con le dimensioni dello schermo) dove vogliamo disegnare un punto.

Un'altra utilissima istruzione, di natura un po' misteriosa, e' la preset. Per definire le sue funzioni diciamo che essa e' il negativo grafico della PSET. Al contrario della PSET che attiva il pixel con il colore di scrittura, la preset attiva il pixel con il colore del fondo su cui si scrive. L'effetto grafico contingente e' quindi nullo in ogni caso, anche se noi operassimo in modo "reverse", ossia ponendo nel programma un'istruzione color 7,0 che- lo anticipiamo- fa passare il video con scrittura in nero (0) su fondo bianco (7).

A che cosa serve la PRESET, allora? Con i pochi mattoni grafici che abbiamo finora fabbricato non e' semplice spiegarlo. Intanto e' immediato osservare che se, dopo aver acceso un pixel con una PSET(x,y) volete spegnerlo dovrete usare una PRESET(x,y) di stesse coordinate.

Ma l'impiego piu' sottile della PRESET potremo vederlo solo insieme ad altre componenti grafiche: esso consente, infatti, di fissare un'origine nascosta di riferimento grafico, senza per altro richiedere l'accensione del pixel alle coordinate corrispondenti. La stessa cosa avrebbe fatto PSET lasciando pero' la traccia del pixel acceso. In tal modo, vedremo, si opera per coordinate relative. Facciamo comunque girare due piccoli ma alquanto astuti programmi per la generazione di una sorta di punteggiate in cui vengono impiegate le istruzioni PSET e PRESET.

Il programma **apix**, qui sotto riportato, comprende alcune istruzioni di cui non vi abbiamo ancora parlato, ma che sono indispensabili per poter gestire piu' facilmente i vari esperimenti.

```
10 'apix:     accensione di pixel
20 CLS:KEY OFF  'in bassa risoluzione
30 SCREEN 1
40 INPUT "premere <CR> per continuare",A
45 CLS:CLEAR
50 INPUT;"X=",X1
60 INPUT "  Y=",Y
70 X=X+1
80 PSET(X,Y)
90 IF X>319 GOTO 40
100 GOTO 70
```

Alludiamo all'istruzione IF, di cui in questa sede diamo solo un breve cenno e che tratteremo ampiamente nel capitolo 5. Ma innanzitutto analizziamo il movimento principale del programma e, per un istante, immaginate che la IF alla linea 90 non esista: anzi, osservate solo le linee 70, 80 e 100; nel loro insieme, quando il programma e' in esecuzione, dopo aver ricevuto i valori di x e y da tastiera, costituiscono una specie di moto perpetuo (finche' M24 e' acceso). Infatti la 70 incrementa il valore di X di un'unita' che trasferisce nel corpo della PSET. Poiche' y rimane costantemente uguale al valore iniziale, per esempio supponiamo y=100, il risultato sara' la generazione di una serie di pixel che giacciono tutti sulla retta orizzontale distante 100 dall'asse delle ascisse, a distanza x dal bordo **sinistro** del video.

Il ciclo evolve fino al raggiungimento della capacita' limite del contenitore associato alla variabile x. Poiche', in bassa risoluzione (screen 1) i pixel con ascissa superiore a 319 sono oltre il confine destro del video, l'attesa che si compia l'evoluzione completa sarebbe inutile in quanto l'effetto grafico residuo sarebbe nullo.

L'unico modo per arrestare le attivita' sarebbe quindi la manovra [CTRL C] alla quale occorrerebbe far seguire il comando RUN e cosi' via. Con l' istruzione IF alla linea 90, invece, tali manovre vengono evitate. Cosa fa la IF? La IF (in inglese significa SE) appartiene alla categoria delle frasi condizionali Basic -che cominciano appunto con la IF- le quali testando una certa condizione (o piu' d'una), consentono di far cambiare la sequenza normale di esecuzione del programma.

Nel nostro caso quando si supera (segno > che significa maggiore) il valore 319, il programma, grazie alla GOTO contenuta nel corpo della stessa IF, ritorna alla linea 40 altrimenti continua la sequenza normale, eseguendo la linea 100. Qui trova la GOTO che, come prima visto, consente l'esercizio ripetuto della PSET alla linea 80. Il salto indietro alla 40, a parte la cancellazione del video per opera della CLS, ci consente di motivare la Input sospensiva contenuta alla linea successiva. Proviamo a immaginare che cosa accadrebbe in assenza di questa Input. Alla prima passata del programma, subito dopo il comando RUN, effettivamente la linea 45 non e' necessaria; ma dopo

si'. Infatti la sospensione del programma subito dopo il primo risultato grafico ci consente di osservarlo comodamente; in caso contrario esso andrebbe immediatamente cancellato dalla CLS (cancella tutto il contenuto del video) alla linea 40.

L'altro programma, di nome **aspix** come riportato nel commento alla linea 10, **accende** e **spegne** **pixel**. Assegnato da tastiera un valore iniziale a x, che corrisponde al primo pixel acceso dalla linea 70, i pixel seguenti vengono accesi per effetto degli incrementi di x eseguiti con le successive assegnazioni di x a carico della linea 60, attivata continuamente dal riciclo operato dalla GOTO della linea 100.

Cio' nondimeno nel corpo del ciclo c'e' anche la PRESET che e' leggermente in ritardo nello spegnimento dei pixel per effetto del fattore M. Se M fosse uguale a zero sarebbe in fase e vedremmo solo un punto che corre in quanto le istruzioni PSET e PRESET hanno effetti opposti. Invece, in virtu' del valore M, anch'esso introdotto da tastiera, possiamo avere alcuni simpatici risultati. Diversamente da apix qui la grafica si produce nella finestra individuata dalla view (figura 4.12).

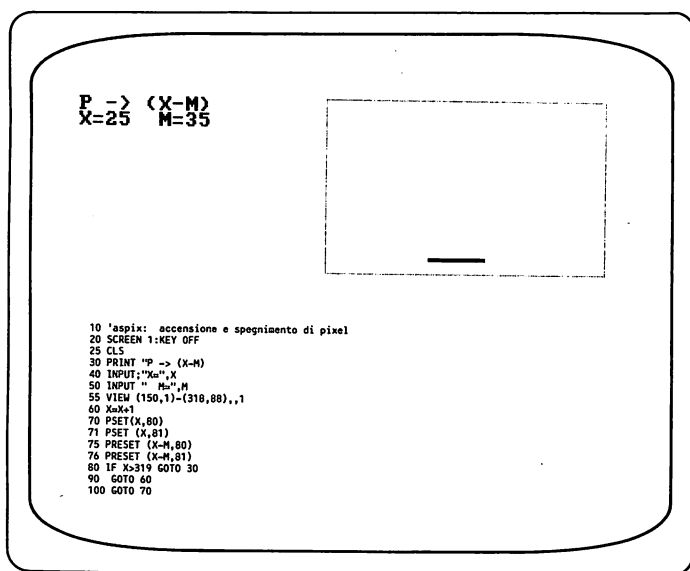



Figura 4.12

Per non complicare il gioco abbiamo limitato l'effetto alle rette Y=80 e Y=81, distanti rispettivamene 80 e 81 pixel dal bordo superiore del video. Nel programma aspix mancano anche le sospensive presenti nel precedente programma apix, in quanto qui l'effetto e' sufficientemente lento e osservabile. L'istruzione di cancellazione non e' stata introdotta per pulire lo spazio grafico, perche' alla funzione provvede la dinamica stessa del programma con la PRESET, quanto per pulire il video nella parte destinata all'introduzione dei dati.

Nel successivo programma **windset2** facciamo pratica di PSET riscrivendo il gruppo di istruzioni di apix in modo pero' da rendere variabile anche y e la pendenza della retta rispetto all'asse x.

```
10 'windset2: punteggiata a pendenza variabile
20 KEY OFF
30 CLS:SCREEN 1
40 WIDTH 20
50 PRINT "P --> (Xo+M),(YO+1)
60 INPUT;"Xo= ",X
70 INPUT;"  Yo= ",Y
80 INPUT;"  M= ",M
100 PRINT
110 VIEW (165,1)-(318,100),,1
120 X=X+M
130 Y=Y+1
140 IF X=320 OR Y=200 GOTO 60
150 PSET (X+50,Y)
160 GOTO 120
```

Anche qui, per non perdere l'immagine grafica, durante lo scorrimento del testo per l'introduzione dei dati, abbiamo usato due finestre come nei precedenti programmi; nella finestra di testo, ridotta a 20 caratteri per riga dalla width di linea 40, introduciamo i vari parametri. I primi dati da introdurre sono Xo e Yo, cioe' le coordinate del punto estremo del segmento: cosi', poiche' la PSET in questo programma aggiunge sempre 50 pixel alla x, se, ad esempio, Xo e Yo sono entrambi nulli, il punto iniziale della retta sara' 50 pixel a destra dell'origine della finestra. Fissato il punto estremo di coordinate (Xo+50,Yo), la scelta del valore di M influenza l'incremento di X operato dalla linea 120, che viene eseguita indefinitamente a causa della GOTO della 160.

La PSET alla linea 150 riceve dunque un valore incrementale di X che dipende da M; al contrario Y cresce solo per effetto della relazione Y=Y+1. In tal modo, modificando il valore di M, possiamo avere rette diversamente inclinate sull'asse X. Infatti per M=0 si ottiene una retta parallela all'asse Y, mentre per M=1 si forma una retta a 45 gradi.

Per M molto grande si ottengono punti che stanno su una parallela all'asse X.

Il programma windset2 (vedi alcune passate nell'hard-copy di figura 4.13) si arresta e torna a chiedere valori di ingresso appena X assume valori che esulano dalla capacita' di rappresentazione orizzontale del video.

Per far cio' alla linea 140 abbiamo piazzato una potente istruzione condizionale vincolata all'esito di un OR logico di due eventi: X uguale a 320 pixel, oppure Y uguale a 200. In ognuno dei due casi il programma, anziche' procedere a eseguire la linea 150, salta daccapo alla linea 60 e si arresta solo con la manovra (CTRL C).

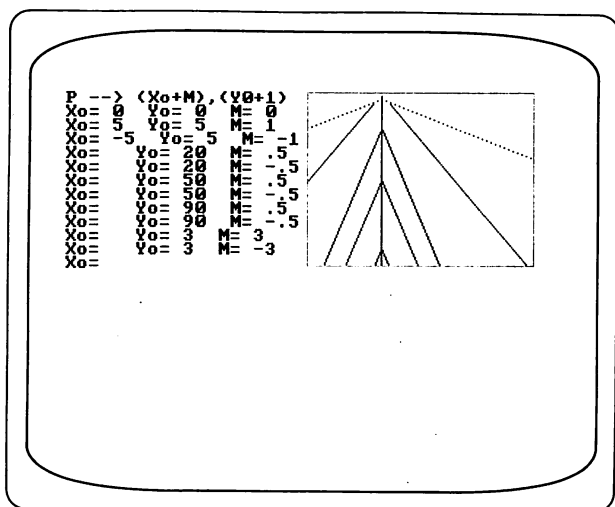



Figura 4.13

La variante **windset8** consente di scegliere il tipo di risoluzione grafica (1,2,3) con la Input di linea 25 e l'altezza della finestra con la Input di linea 38; un'altra novita' e' rappresentata dalla coppia di istruzioni alle linee 170 175, la prima delle quali aspetta un solo carattere da tastiera e la seconda lo analizza. Solo se si tratta del carattere [/] allora il programma torna in testa alla linea 20 consentendo all'operatore di riselezionare il tipo di screen, altrimenti salta indietro alla linea 38 per la scelta dell'altezza della finestra e cosi' via. In figura 4.14 l'hard-copy di una serie di passate di windset8.

```
10 'windset8:      punteggiata a pendenza variabile
20 CLS:KEY OFF'    in finestra proiettiva.
25 INPUT "Risol.= ",Z
30 SCREEN Z
35 WIDTH 20
38 INPUT W
40 IF W<1 OR W>199 GOTO 35
43 VIEW (165,1)-(318,W),,1
50 PRINT "P --> (Xo+M),(Yo+1)
60 INPUT;"Xo= ",X
70 INPUT;"  Yo= ",Y
80 INPUT;"  M= ",M
100 PRINT
115 PSET (X+50,Y)
120 X=X+M
130 Y=Y+1
140 IF X>640 OR Y>400 GOTO 170
160 GOTO 115
170 W$=INPUT$(1)
175 IF W$="/" THEN 20
180 GOTO 38
```

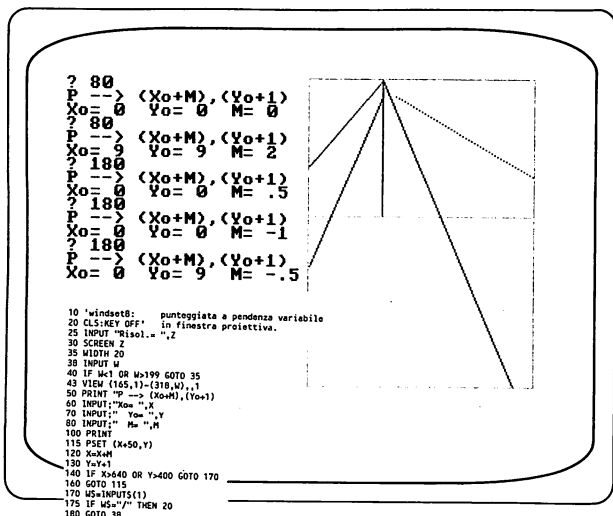



Figura 4.14

4.4.2 LA RETTA

Con la retta il discorso comincia veramente a farsi interessante.
Infatti nel repertorio delle istruzioni grafiche di M24 ne esiste una che fabbrica direttamente la retta dichiarando le coordinate di due punti. Nella sua espressione piu' elementare, l'istruzione, denominata **line**, ha il seguente formato:

```
line(X1,Y1)-(X2,Y2)
```

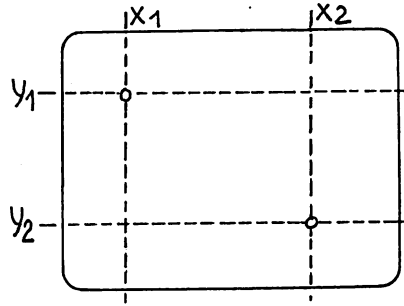



Figura 4.15

Introducendo le coordinate dei punti P1=(50,100) e P2=(150,200), la line diventa:

```
line(50,100)-(150,200)
```

e si ottiene sul video la retta passante per P1 e P2 (vedi figura 4.15).

Per sperimentare varie combinazioni di punti vediamo il seguente programma (con relativa figura 4.16) denominato **linvar.**

Anche questo programma, come i precedenti, utilizza la tecnica di muovere i parametri di un'istruzione per valutare gli effetti globali sul risultato. L'ambiente grafico scelto e' a bassa risoluzione (screen 1). I dati di Input alle linee 30, 40 e 50 sono l'ascissa di un estremo del segmento e le coordinate dell'altro. L'ordinata Y1 e' stata assunta costante e pari a 100.

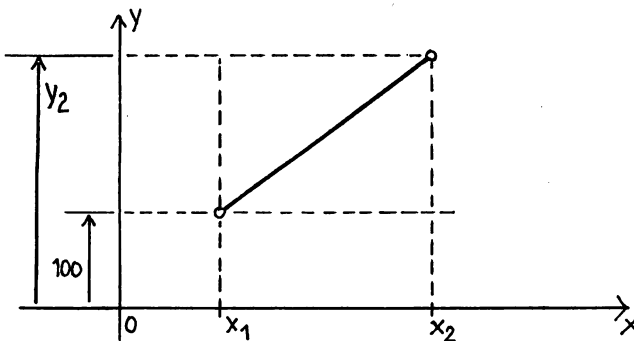



Figura 4.16

Se vogliamo tenere costante anche X1 si potra' costruire una stella di rette passanti per P1.

Una variante possibile di linvar sfrutta l'esperienza dei programmi tipo apix e consente di tracciare automaticamente una stella di rette uscenti da un punto X1 Y1. A tal fine basta variare i valori di X2 Y2 come in figura 4.17.

Se muoviamo soltanto Y2, a parita' di X2, l'estremo mobile del segmento sara' costretto a percorrere la retta X=X2 parallela all'asse delle ordinate (vedi figura 4.18). Se, invece, muoviamo X2 a parita' di Y2, l'estremita' del segmento e' vincolata alla retta Y=Y2 parallela all'asse delle ascisse (vedi ancora figura 4.18).

Un programma capace di comportarsi nel modo sopra descritto dovra' ricevere nel primo caso il valore di X2, nel secondo caso quello di Y2. Consideriamo prima il caso dell'estremo vincolato alla retta X=X2 (programma **linvary**). L'istruzione generatrice della stella sara':

```
line(X1,100)-(X2,Y)
```

L'altro programma di nome **linvarx** avra', invece, un'istruzione del tipo:

```
line(X1,100)-(X,Y2)
```

Ovviamente, in entrambi i casi, dovremo provvedere a fornire i successivi valori alle variabili in gioco. A tal fine sono state inserite le opportune istruzioni per fare iterare la line. I programmi linvary e linvarx, qui sotto riportati, si fermano solo al raggiungimento del valore massimo delle variabili. Siccome si tratta di variabili in semplice precisione tale valore e' pari a 3.40282E+38: conviene, quindi, interromperli con la manovra (CTRL+Break).

```
10 'linvary:segmento variabile
20 CLS:KEY OFF:SCREEN 2
25 WINDOW (0,0)-(639,199)
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X2
55 Y=Y+3:LINE (X1,100)-(X2,Y)
75 GOTO 55
```

```
10 'linvarx:segmento variabile
20 CLS:SCREEN 3:KEY OFF
30 INPUT; "X1= ",X1
40 INPUT; "  Y2= ",Y2
55 X=X+5
60 LINE (X1,100)-(X,Y2)
75 GOTO 55
```

Se facciamo variare anche Y2 nella line di linvary otteniamo il programma **linvaryx**.

```
10 'linvaryx:segmento variabile
20 CLS:SCREEN 3:KEY OFF
25 WINDOW (0,0)-(639,399)
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X
50 INPUT;  "  Y2= ",Y
55 Y=Y+5
56 X=X+1
60 LINE (X1,100)-(X,Y)
65 GOTO 55
70 INPUT " *  premi <CR> per continuare    ",A
80 GOTO 20
```

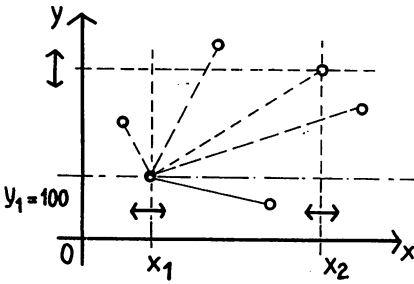



Figura 4.17

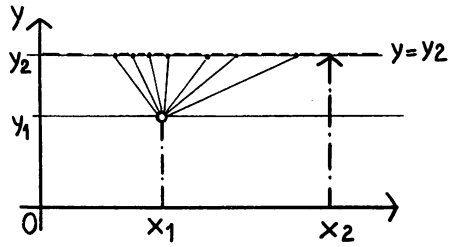
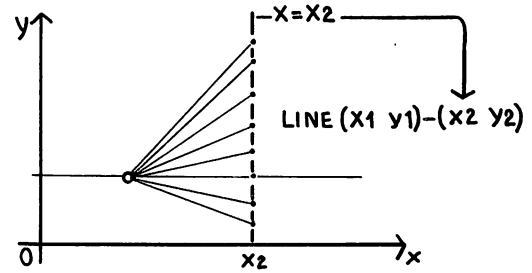



Figura 4.18

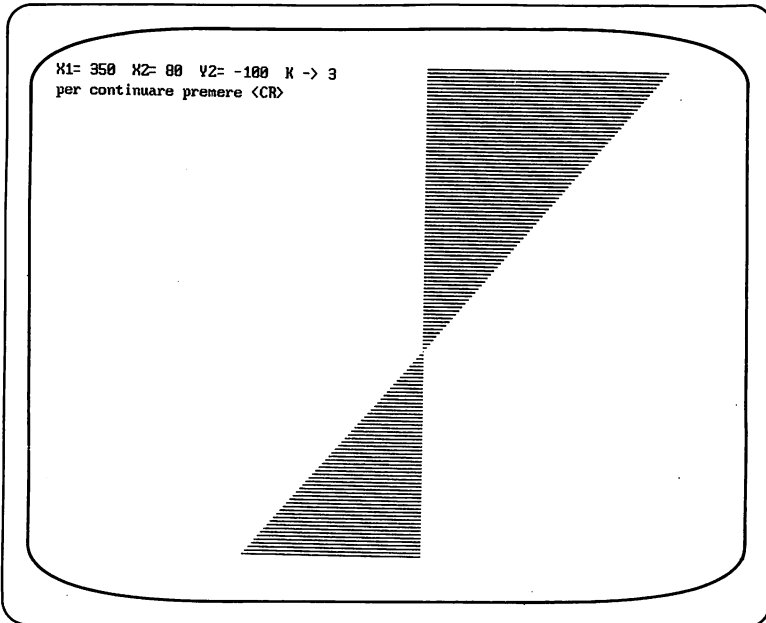



Figura 4.19

Con l'introduzione di una variabile K nella 55 e 56 otteniamo anche di far evolvere i grafici a velocita' controllata (programma invaryx1 e invaryx2):

```
10 'invaryx1:segmento variabile
20 CLS:SCREEN 3:KEY OFF
25 WINDOW (0,0)-(639,399)
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X
50 INPUT;  "  Y2= ",Y
52 INPUT;"  K -> ",K
55 Y=Y+K
56 X=X+K
60 LINE (X1,100)-(X,Y)
65 GOTO 55
```

```
10 'invaryx2:segmento variabile
20 CLS:SCREEN 3:KEY OFF
25 WINDOW (0,0)-(639,399)
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X
50 INPUT;  "  Y2= ",Y
52 INPUT"  K -> ",K
55 Y=Y+K
56 X=X+K
60 LINE (X1,Y)-(X,Y)
62 IF X>3000 GOTO 70
65 GOTO 55
70 INPUT"per continuare premere <CR>",W
75 GOTO 20
```

In figura 4.19 forniamo il risultato grafico ottenuto in hard-copy dal programma invaryx2.

Vediamo ora un'ulteriore possibilita' dell'istruzione line: si tratta della variante ottenuta aggiungendo l'appendice **STEP** alla parola chiave **line.** Spieghiamola con il seguente programma di nome **lindy** e l'annessa figura 4.20:

```
10 'lindy: segmento a pendenza variabile
20 '        con estremita' legata a PSET
30 CLEAR:CLS:SCREEN 3:KEY OFF
35 WINDOW (0,0)-(639,399)
40 INPUT; "Xo=",XO
50 INPUT; "  Yo=",YO
60 INPUT; "  Dy=",DY
70 PSET(XO,YO)
80 LINE - STEP(200,DY)
90 INPUT "",A
100 GOTO 30
```

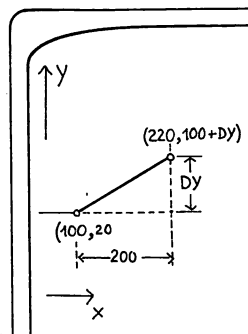



Figura 4.20

Come potete notare, rispetto a una line normale, nella line step mancano le coordinate di un estremo del segmento da disegnare. Il motivo di cio' e' che a fornirle provvede l'istruzione PSET, le cui coordinate costituiscono il riferimento per la line step stessa. Allora, nel programma lindy, dopo aver impostato Xo e Yo che servono alla PSET, bisogna fornire alla line step il parametro DY che, come risulta dalla figura 4.20, va inteso come un incremento rispetto a Yo.

La line step, dunque, permette di concatenare tra loro segmenti diversi fino a costruire figure chiuse, come appare nel programma **lindy2**, che costruisce un triangolo con i dati in ingresso Xo=20 Yo=100 Dy=-50 (vedi figura 4.21).

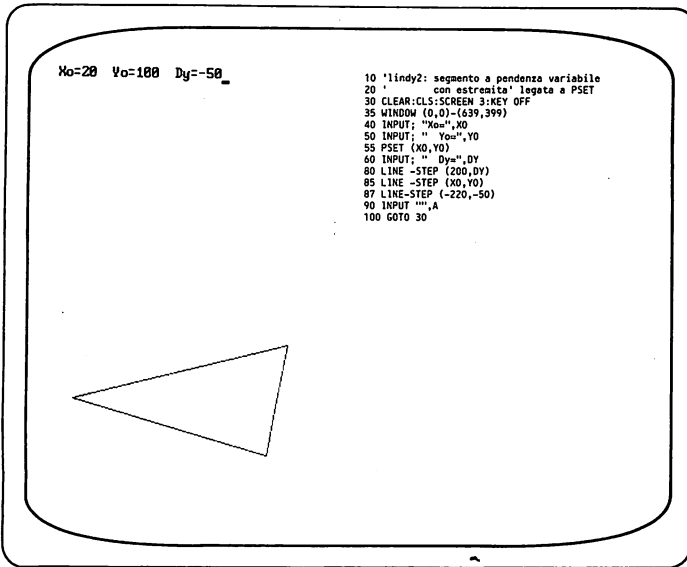



Figura 4.21

4.4.3 IL RETTANGOLO

Una variante interessante della line consente, date le coordinate degli estremi di un segmento, di costruire un rettangolo avente per diagonale il segmento stesso. Piu' precisamente basta scrivere:

```
line(50,100)-(150,200),,B
```

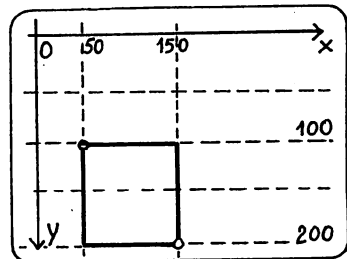



Figura 4.22

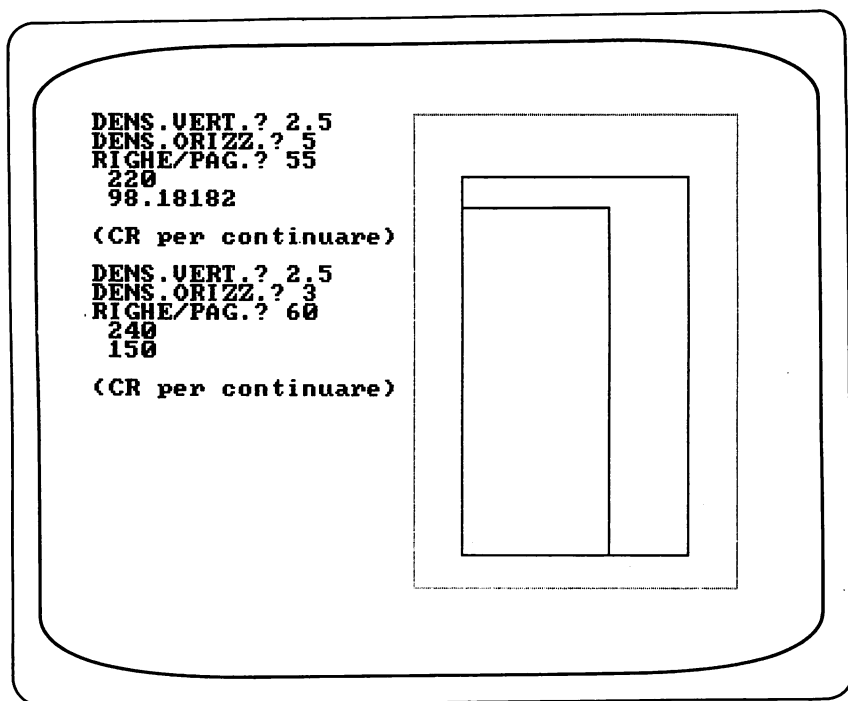



Figura 4.23

```
10 'gabline: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina. Controllo di impaginazione.
40 KEY OFF:CLS
50 SCREEN 1:WIDTH 19
60 INPUT "DENS.VERT.";DV
70 INPUT "DENS.ORIZZ.";DO
80 INPUT "RIGHE/PAG.";RP
90 CR=2700/RP
100 LG=10*CR/DO
110 HG=10*RP/DV
120 PRINT HG
130 PRINT LG
140 WINDOW (0,0)-(210,297)
150 VIEW (160,1)-(318,198),,1
170 LINE (30,20)-(30+LG,20+HG),,B
200 PRINT
210 INPUT "(CR per continuare)",A
220 PRINT:GOTO 60
```

E' stato sufficiente, quindi, aggiungere qualche virgola e la lettera B (BOX) all'istruzione per disegnare un rettangolo come in figura 4.22.

Se, nell'istruzione, aggiungiamo anche una F (fill), l'effetto sara' di colorare l'area racchiusa dal rettangolo. Segue ora una variazione del noto programma gabbia -che ridefiniamo **gabline-** nel quale inseriamo una line con la suddetta variante B.

In tale programma i risultati LG e HG vengono aggiunti come incrementi DX e DY, rispettivamente a X2 e Y2 della line; il risultato e' quello di figura 4.23 nella quale si riporta anche il listato di gabline.

Nella suddetta figura appare anche un rettangolo (disegnato con l'istruzione view alla linea 150 di gabline) che serve da riferimento, come una pagina di carta in cui deve trovar spazio il testo scritto. Per inciso le coordinate assolute rispetto alla quali tale pagina si rapporta sono proprio le dimensioni di un foglio A4: con tali valori e' stata definita la window alla linea 140.

Nella figura 4.24 riportiamo, per comodita', gli estremi delle coordinate utilizzate per costruire sia la cornice della window sia la stessa gabbia il cui vertice inferiore sinistro ha coordinate relative (30,20). Ovviamente, le coordinate del vertice Q opposto al punto P sono indicate come funzioni di LG e HG.

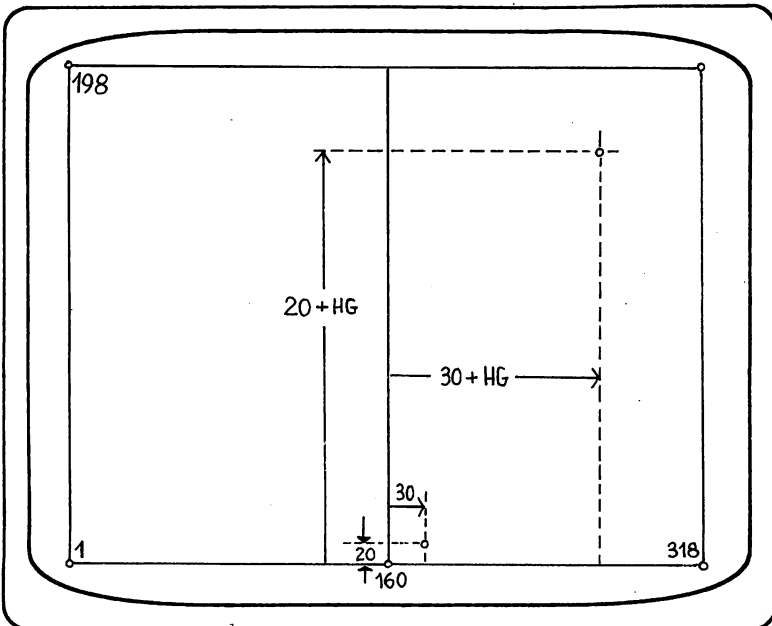



Figura 4.24

Segue, infine, un'ultima elaborazione del programma gabbia, denominato **fibline**, che utilizza un'istruzione line con l'opzione B (BOX).

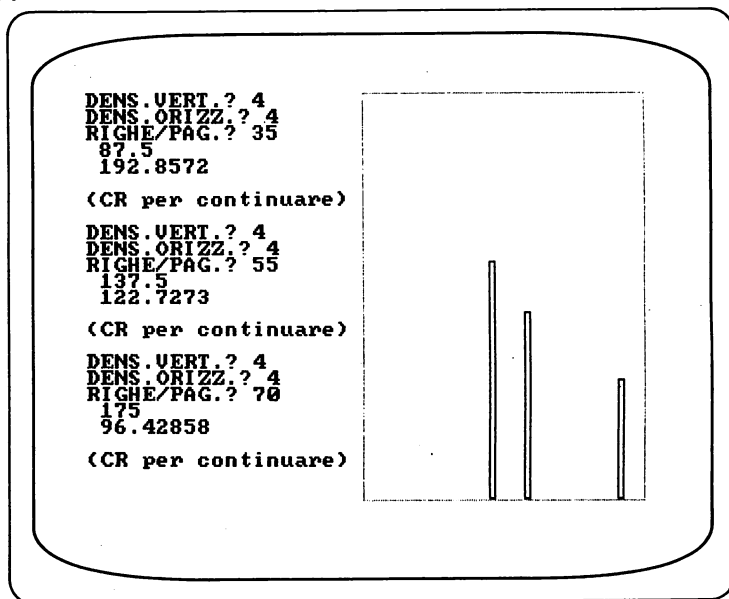
```
10 'fibline: calcolo al variare delle densita' verticale
20 '        e orizzontale e del numero di righe per pa-
30 '        gina. Controllo di impaginazione.
40 KEY OFF:CLS
50 SCREEN 1:WIDTH 19
60 INPUT "DENS.VERT.";DV
70 INPUT "DENS.ORIZZ.";DO
80 INPUT "RIGHE/PAG.";RP
90 CR=2700/RP
100 LG=10*CR/DO
110 HG=10*RP/DV
120 PRINT HG
130 PRINT LG
140 WINDOW (0,0)-(210,297)
150 VIEW (160,1)-(318,198),,1
170 LINE (LG-2,0)-(LG+2,HG),,B
200 PRINT
210 INPUT "(CR per continuare)",A
220 PRINT:GOTO 60
```


Figura 4.25

In figura 4.25 si trova l'hard-copy dei grafici ottenuti variando solo RP e lasciando inalterati DO=DV=4.

Per costruire un rettangolo con i soli valori di HG e LG e' stato necessario un piccolo artificio, come illustrato nella seguente figura 4.26:

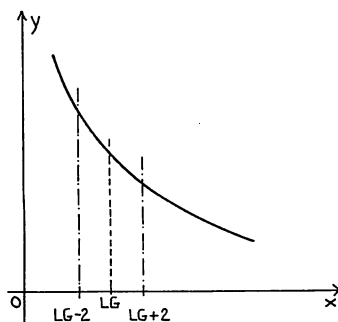



Figura 4.26

Se, invece di costruire una fascia costante intorno a un punto e corrispondente al rettangolo che ha per base XA-XB=(LG+2)-(LG-2) e per altezza HG, si vuol rendere variabile e selezionabile all'esterno anche la base di tale fascia, basta sostituire alla costante 2 una variabile F e muoverla con una INPUT "F= ",F.

4.4.4 LA CIRCONFERENZA E L'ELLISSE

Nella sua espressione piu' semplice l'istruzione che traccia una circonferenza ha il seguente formato:

```
CIRCLE(X,Y),R
```

dove X e Y sono le coordinate del centro e R il raggio.

Dobbiamo notare che la misura del raggio dichiarata nell'istruzione non puo' coincidere con un numero di pixel, in quanto, come gia' osservato in altre occasioni, la distanza verticale tra due pixel e' maggiore di quella tra due pixel in orizzontale. L'istruzione CIRCLE compensa automaticamente la disuguaglianza tra queste due distanze consentendo di tracciare delle circonferenze perfette.

Quando si voglia alterare questa automatica compensazione e ottenere un'ellisse e' sufficiente scrivere un numero nella circle, come qui di seguito indicato:

```
CIRCLE(X,Y),R,,N
```

dove per N=0.8 si ottiene un cerchio, per N < 0.8 un'ellisse con semiasse maggiore orizzontale, e per N > 0.8 quella a semiasse maggiore verticale. Nella figura 4.27 riportiamo una famiglia di circonferenze concentriche e, qui di seguito, il relativo programma, di nome **fincer**, che l'ha generata.

```
10 'fincer: famiglia di circonferenze
20 '        su due finestre
30 KEY OFF:CLS
40 INPUT "Risoluz.=",Z
50 SCREEN Z
55 WIDTH 15
60 WINDOW (0,0)-(639,399)
65 VIEW (150,1)-(318,198),,1
70 INPUT "X= ",X
80 INPUT "Y= ",Y
100 R=0
110 R=R+10
120 CIRCLE(X,Y),R
130 IF R>120 GOTO 150
140 GOTO 110
150 INPUT"",W
160 GOTO 60
```

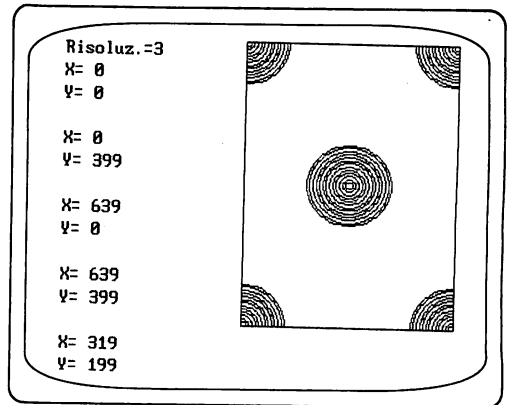



Figura 4.27

CAPITOLO
CINQUE

ALTRE FRASI NOTEVOLI DEL BASIC

In questo capitolo tratteremo i cicli iterativi, le frasi logiche, il controllo dei programmi e dei sottoprogrammi e i cambiamenti di scala: quasi tutto, naturalmente, in chiave grafica. Questi nuovi "mattoni", che aggiungeremo alla nostra conoscenza Basic, ci permetteranno di migliorare notevolmente i nostri obiettivi informatici.

Sono stati inseriti nel capitolo vari programmi e relative hard-copy, ma il consiglio e' che mettiate alla prova la vostra creativita' direttamente su M24.

5.1 CICLI ITERATIVI

Nella vita quotidiana siamo immersi in situazioni di tipo periodico. Gli esempi piu' comuni sono l'alternarsi del giorno e della notte, il succedersi delle stagioni, il ciclo dei giorni della settimana. In un elaboratore come il nostro, dotato di soli organi di entrata e di uscita tradizionali, come la tastiera e il video (ma potremmo attrezzarci diversamente...), la ripetitivita' viene rivolta a situazioni create artificialmente per scopi particolari. Per motivi di chiarezza, in questo capitolo mostreremo soltanto impieghi elementari di queste proprieta' cicliche.

Esempi di iterazione li abbiamo per altro gia' visti in programmi come apix (paragrafo 4.4.1), in cui il processo ciclico viene realizzato con l'uso della GOTO e della relazione di assegnazione X=X+1; ricordiamo che per arrestare tale processo, senza la manovra CTRL Break, e' stato necessario inserire all'interno del programma una frase condizionale, che agisse come un interruttore al verificarsi di un certo evento.

Nel caso specifico la condizione scelta e': "quando l'ascissa supera il valore 319" (IF X > 319), "vai alla linea 40" (GOTO 40). La stessa cosa e' realizzabile con una coppia di istruzioni (**FOR...NEXT**), che racchiudono l'operazione che si desidera ripetere. Riprendiamo il programma apix e analizziamo il suo diagramma logico (figura 5.1).

Come si vede la condizione logica viene esaminata dalla IF che fa deviare il programma verso il **SI** quando l'evento e' soddisfatto, oppure verso il **NO**, riciclando alla linea 70.

Con il programma **forapix** si ottiene lo stesso effetto, ma con le istruzioni FOR alla linea 70 e NEXT alla 90. Questa coppia di istruzioni serve a far ripetere per un certo numero di volte un determinato gruppo di istruzioni che costituiscono il **corpo** della "FOR...NEXT".

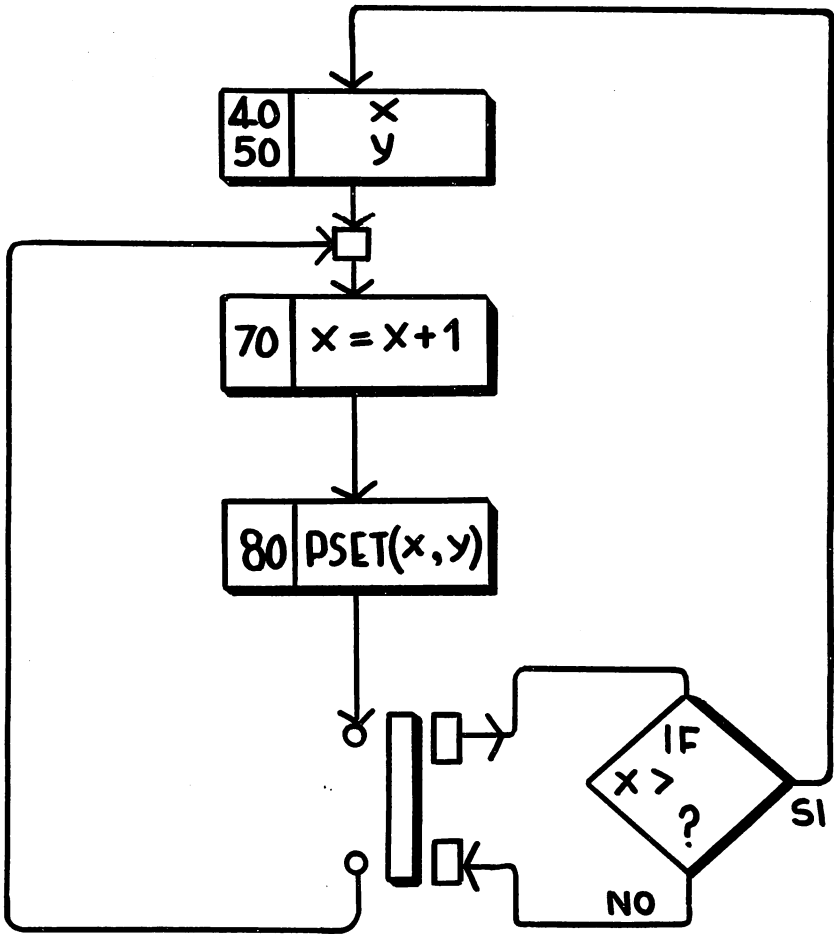
Per far questo l'istruzione deve avere nel suo interno un meccanismo che conta quante volte il corpo delle istruzioni debba riciclare.

```
10 'apix:     accensione di pixel
20 CLS:KEY OFF  'in bassa risoluzione
30 SCREEN 1
40 INPUT "premere <CR> per continuare",A
45 CLS:CLEAR
50 INPUT;"X=",X1
60 INPUT "  Y=",Y
70 X=X+1
80 PSET(X,Y)
90 IF X>319 GOTO 40
100 GOTO 70
```

Figura 5.1

```
10 'forapix: accensione di pixel
20 KEY OFF  'in bassa risoluzione
30 SCREEN 1
40 INPUT "premere <CR> per continuare",A
45 CLS
50 INPUT;"X=",X1
60 INPUT "  Y=",Y
70 FOR X=X1 TO 319
80 PSET(X,Y)
90 NEXT X
100 GOTO 40
```

E' cio' che avviene con una opportuna **variabile di controllo** dichiarata insieme alla FOR. A tale variabile bisogna assegnare un valore limite inferiore e uno superiore.
L'incremento, o passo, che viene dato automaticamente dalla macchina a ogni ciclo e' uguale 1. Una variante della FOR prevede di assegnare al passo un valore qualsiasi positivo o negativo, **purche' diverso da zero.** In tal caso, nell'istruzione FOR, occorre utilizzare l'appendice **STEP** (passo) e assegnargli un valore.
In generale, dunque, se chiamiamo K la variabile di controllo, la sintassi della FOR, con riferimento alla figura 5.2, e' la seguente:

```
FOR K=X1 TO X2              STEP P

[CORPO]

NEXT K   (prossimo valore di K)
```

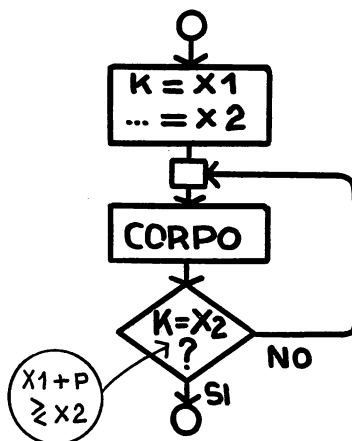



Figura 5.2

Torniamo al programma forapix per alcune considerazioni. Prima di tutto notiamo che, rispetto al programma apix, e' cambiato qualcosa anche nella frase INPUT alla linea 70; infatti qui la variabile alla quale assegniamo il valore si chiama X1.
Cio' nondimeno nell'argomento della PSET alla linea 80 viene dichiarata una variabile X. E, inoltre, la variabile di controllo e' proprio la X.
Potreste chiedervi: "E' questo l'unico modo di gestire le FOR... NEXT?"

Per usare la FOR...NEXT occorre ricordare i seguenti punti:

1. La variabile di controllo della FOR...NEXT deve apparire in **almeno una** delle istruzioni che costituiscono il corpo da riciclare, se vogliamo che ad ogni ciclo esso risulti affetto da una certa variazione.

2. Il limite inferiore e il limite superiore non devono coincidere.

3. I limiti possono essere delle variabili **esterne al ciclo.**

4. Quando e' espresso, anche il passo puo' essere una variabile esterna.

5. E' possibile **annidare** una FOR...NEXT dentro un'altra (vedi figura 5.3).

6. Non e' necessario dichiarare nella NEXT il nome della variabile di controllo. In caso di piu' NEXT il Basic ricicla la variabile dichiarata nella FOR...NEXT immediatamente precedente.

7. Non e' consentito intrecciare due FOR...NEXT, come in figura 5.4 (l'effetto non e' dannoso ma non da' risultati corretti); non e' consentito neppure assegnare uno STEP=0, perche' verrebbe attivato un ciclo infinito.

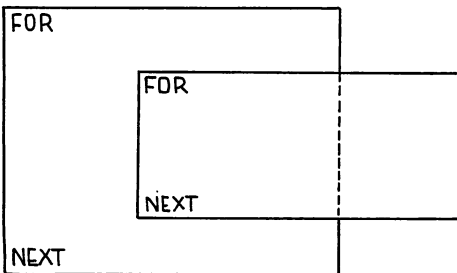



Figura 5.3

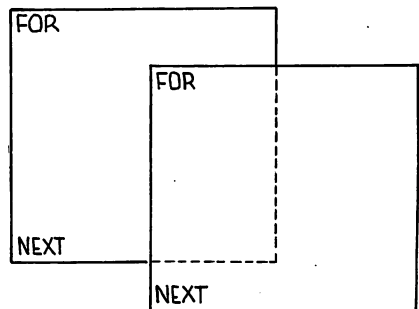



Figura 5.4

L'interpretazione di forapix ora dovrebbe essere completamente chiara. La perplessita' sollevata sulla coincidenza della variabile di controllo X con le variabili della PSET cade per quanto detto al precedente punto 1. Tale possibilita' e' contemplata nella sintassi dell'istruzione ed e' quindi perfettamente lecita, come appare nei seguenti programmi, denominati **forapix2** e **forset**, nei quali **non** e' necessario rinominare la variabile esterna come, per chiarezza, abbiamo fatto per X nel programma forapix1.

```
10 'forapix1: accensione di pixel
20 KEY OFF  'in bassa risoluzione
30 SCREEN 1
40 INPUT "premere <CR> per continuare",A
45 CLS
50 INPUT;"X=",X1
60 INPUT;"  Y=",Y
65 INPUT "  passo --> ",P
70 FOR X=X1 TO 319 STEP P
80 PSET(X,Y)
90 NEXT X
100 GOTO 40
```

```
5 'forset:   accensione
6 'e spegnimento di pix.
10 CLS:SCREEN 1:KEY OFF
12 INPUT Y
13 INPUT Z
15 FOR X=0 TO 500 STEP Z
20 PSET (X,80)
25 PRESET (X-Y,80)
30 NEXT
35 INPUT ;"",A
40 GOTO 10
```

Anzi, e' addirittura possibile sostituire ai limiti inferiore e superiore, nonche' allo STEP, un'espressione simbolica contenente sia le variabili esterne, sia la stessa variabile di controllo, come e' dimostrabile nel seguente **forapix2.**

```
10 'forapix2: accensione di pixel
20 KEY OFF:  'in bassa risoluzione.
30 SCREEN 1
40 INPUT;"premere <CR> per continuare",A
45 CLS
50 INPUT;"X=",X
60 INPUT;"  Y=",Y
65 INPUT "  passo --> ",P
70 FOR X=X*5 TO 319-1.5*Y STEP P/Y
80 PSET(X,Y)
90 NEXT X
100 GOTO 40
```

Un ultimo rilievo riguarda l'impiego anomalo, ma utile, della FOR..NEXT, cioe' quello di usarla come un'attesa programmata. E' sufficiente scrivere una FOR...NEXT senza corpo, con un numero di passi da fare che dipende dall'attesa che vogliamo relizzare in un certo punto del programma. Ad esempio, per un'attesa di circa 5 secondi, su M24 dovremo assegnare a una FOR...NEXT senza corpo 6000 passi da 1.

Il programma **forat,** che qui riportiamo, prevede un certo dialogo con l'operatore:

```
10 'forat: attesa programmata
20 CLS:SCREEN 1:KEY OFF
22 M=1/2620
25 INPUT"vuoi modificare la taratura? (si=1)     ---> ",A
26 IF A=1 THEN INPUT" m= ",M
30 INPUT"tempo in secondi --> ",T
40 FOR K=1 TO T STEP M
50 NEXT
60 PRINT:PRINT "sono trascorsi"T"secondi
65 INPUT"per continuare premi <CR>",C
70 GOTO 20
```

Si noti che alla linea 22 il programma assegna automaticamente il valore del passo M=1/2620. Tale valore, essendo molto piu' piccolo di uno, consente di assegnare un numero di cicli tale da dilazionare la fine dell'iterazione di un tempo corrispondente agli effettivi secondi trascorsi da quando e' stato premuto [CR]. In quel preciso istante la macchina assegna il valore di T introdotto nella FOR scritta alla linea 40; piu' precisamente definisce in essa il limite superiore della variabile di controllo. Al termine dei T cicli la macchina esce dell'iterazione e stampa il valore di T. Questo valore non ha tanto il significato di un risultato -infatti ci era ben noto perche' l'avevamo comunicato noi alla macchina- quanto perche', nel momento in cui appare sul video, abbiamo la sensazione del tempo trascorso.

Il programma forat consente inoltre, attraverso la richiesta formulata alla linea 25 (si=1), di modificare la taratura della FOR...NEXT. Se, infatti, rispondiamo 1, la IF alla linea successiva se ne accorge e ci propone allora (THEN) quale altro valore di M vogliamo introdurre.

Alla linea 65 troviamo la consueta INPUT sospensiva per consentirci di leggere quanto la PRINT ha stampato alla linea precedente.

Sulla meccanica della FOR...NEXT dobbiamo infine osservare che, rispetto alla IF [condizione] GOTO, essa consente una maggiore chiarezza espositiva.

Orbene, a conclusione di questa sorta di trattato sui cicli iterativi, non possiamo non accennare a un'altra coppia di istruzioni, per certi aspetti piu' interessante della FOR...NEXT. Si tratta delle **WHILE...WEND**, che, come le FOR...NEXT, sono istruzioni terminali comprendenti un corpo di istruzioni da ripetere ciclicamente. Questa volta, pero', il numero delle iterazioni non dipende da un limite superiore, ma solo dal verificarsi di una certa condizione. La sintassi e' delle piu' semplici.

Finche' [WHILE] sussiste una certa [CONDIZIONE], continua a eseguire il gruppo di [ISTRUZIONI] che precedono la [WEND].

Nella sua logica rassomiglia alla IF, anche se questa ha generalmente lo scopo di far eseguire azioni diverse a seconda che una certa condizione sussista o no. Un confronto tra IF e WHILE appare utile attraverso l'analisi di uno stesso programma gia' visto in tema di **line** al paragrafo 4.4.2. Si tratta del programma linvary che riprendiamo nella sua forma originale.

```
10 'linvary:segmento variabile
20 CLS:KEY OFF:SCREEN 2
25 WINDOW (0,0)-(639,199)
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X2
55 Y=Y+3
60 LINE (X1,100)-(X2,Y)
75 GOTO 55
```

Avrete notato che in esecuzione le rette uscenti dal punto prescelto (X1,X2) hanno un'estremita' che si appoggia alla retta X=X2 parallela all'asse delle ordinate.

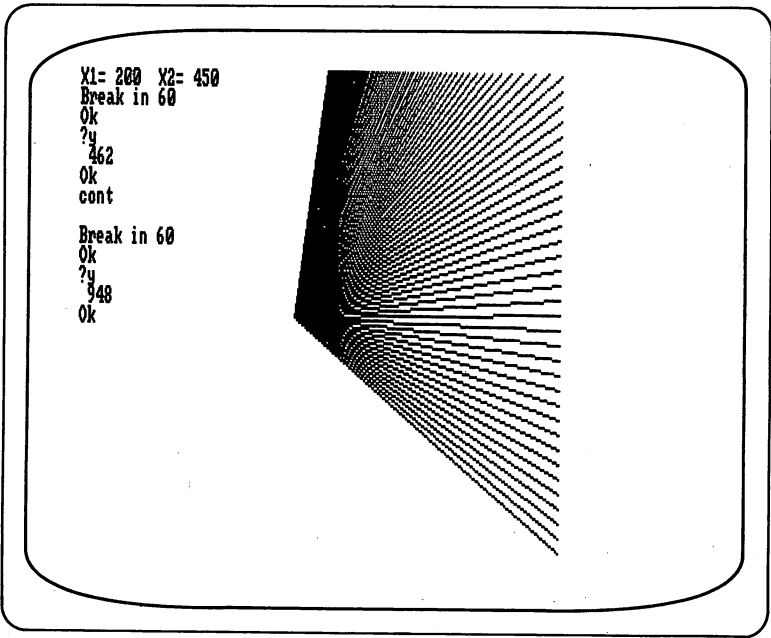



Figura 5.5

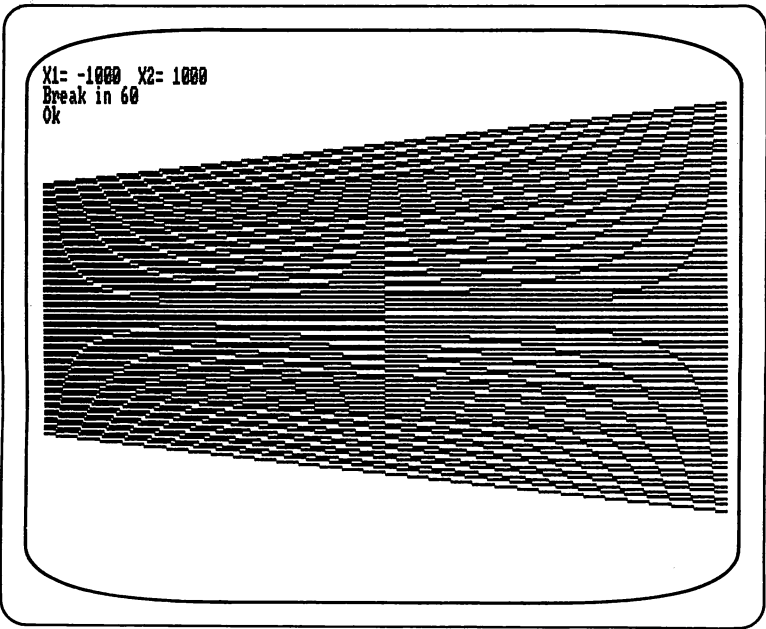



Figura 5.5a

Ad esempio per X1=200 e X2=450, si e' ottenuto il grafico di figura 5.5, **interrotto** in fase di lenta evoluzione quando Y non aveva ancora raggiunto il valore massimo consentito dall'espressione intera dei numeri, cioe' 32767. Per curiosita', facciamo osservare che il grafico e' stato interrotto dopo appena un minuto dal suo lancio, mentre avremmo dovuto aspettarne 8 perche' il ciclo si estinguesse naturalmente. Cio' da' la sensazione materiale, anche se grossolana, di distanza tra un punto di una retta e il suo punto all'infinito. Altri effetti di linvary sono riportati nella figura 5.5a.

Volendo interrompere il ciclo in modo programmato, anziche' con la manovra [CTRL Break] possiamo scrivere un programma, ad esempio, con una WHILE...WEND sensibile al verificarsi di una certa condizione.

Ne possiamo inventare infinite di condizioni: quella che abbiamo scelto consente anche di vedere quale genere di condizioni possiamo scrivere con una IF o una WHILE.

I due seguenti programmi (linvary1 con la IF e linwary con la WHILE) realizzano lo stesso risultato di arrestare su condizione il grafico prima illustrato.

```
10 'linvary1:segmento variabile
20 CLS:SCREEN 2:KEY OFF
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X2
55 Y=Y+5
56 IF Y>50+(X2-X1)*1.5 GOTO 80
60 LINE (X1,100)-(X2,Y)
75 GOTO 55
80 INPUT"",A
90 GOTO 20
```

```
10 'linwary:segmento variabile
20 CLEAR:CLS:SCREEN 2:KEY OFF
30 INPUT; "X1= ",X1
40 INPUT; "  X2= ",X2
52 WHILE Y<50+(X2-X1)*1.5
55 Y=Y+5
60 LINE (X1,100)-(X2,Y)
80 WEND
82 INPUT"",A
85 GOTO 20
```

Seguono esempi di generazione di grafiche con la FOR...NEXT, come i programmi **tapfor** e **cervar1**.

```
10 'tapfor: esempio di "for"
20 '        annidate (p=24)
25 CLS:SCREEN 1:KEY OFF
30 '
35 LOCATE 1,1
40 INPUT;"passo (>15)-->",P
50 IF P<15 GOTO 35
60 CLS
70 FOR A=1 TO 20
80 '
90 FOR Y=255 TO 0 STEP -P
100 LINE (0,Y-A)-(511,Y-A)
110 NEXT
120 '
130 NEXT
140 '
150 INPUT"",W
160 GOTO 20
```

```
10 'cervar: famiglia di circonf.
20 CLS:KEY OFF:INPUT "Risol.=",Z
30 IF Z=0 THEN 20
40 SCREEN Z
50 WINDOW (0,0)-(319,199)
60 VIEW (135,1)-(318,198),,1
70 INPUT "passo--> ",Z
80 IF Z=0 THEN 70
90 FOR R=1 TO 200 STEP Z
100 CIRCLE(162,128),R
110 NEXT
120 INPUT"",W
130 GOTO 20
```

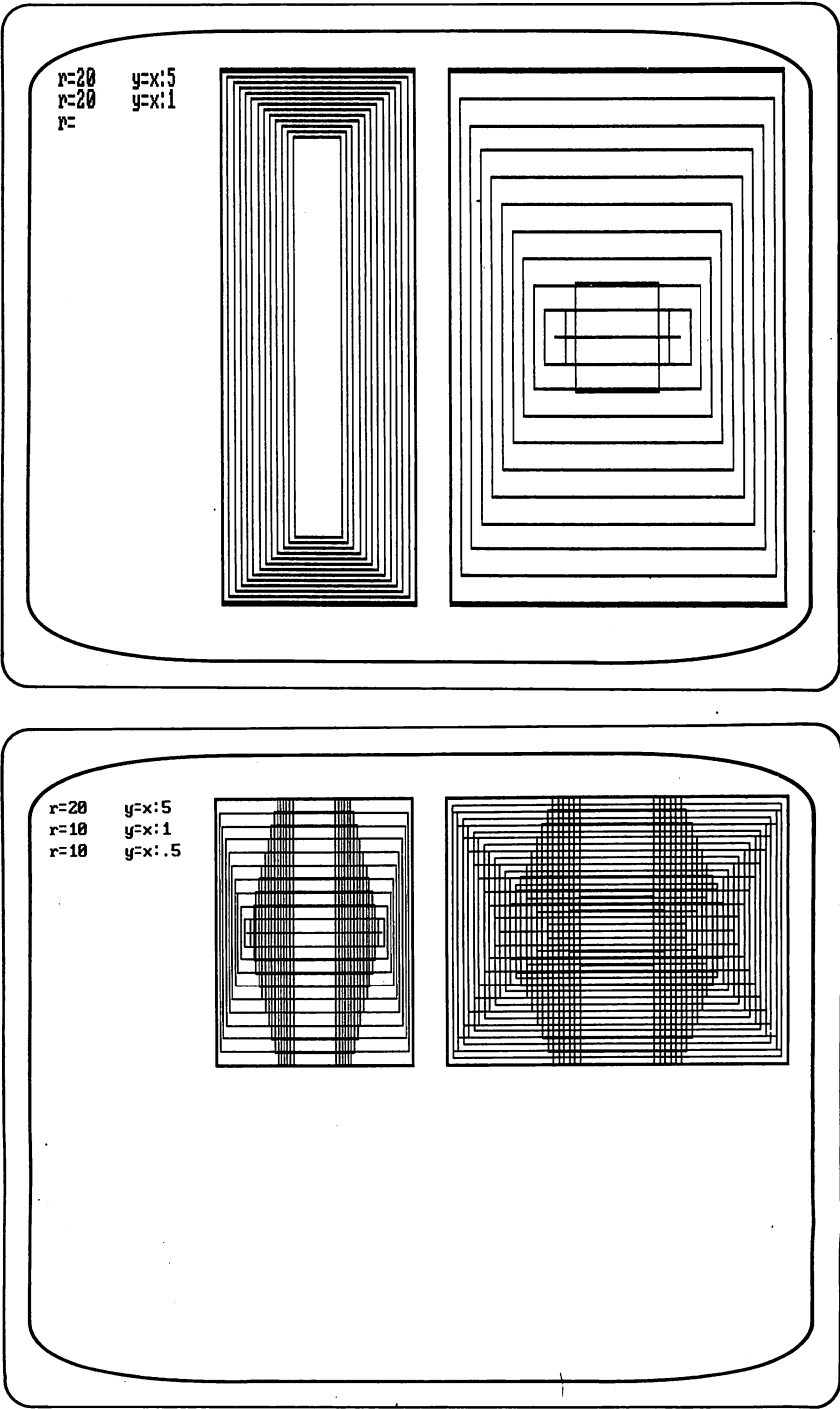



Figura 5.6

Riportiamo ora il programma **ragfor** e tre sue varianti. L'obiettivo e' di generare delle cornici di passo **r** tramite i rettangoli generati da una **line**.

Nella versione base tali cornici vengono generate complete solo se si e' scelta l'alta risoluzione, in quanto la line alla linea 70 e' tarata per coordinate video che vanno oltre i valori consentiti dalle risoluzioni piu' basse (screen 1 e 2).

Nella prima variante (ragfor1) le cornici, anziche' essere generate a pieno video, appaiono in una finestra definita dalla linea 45, preceduta dalla window di linea 42 per far si' che la finestra riproduca l'intero intervallo grafico. (Provate infatti a neutralizzare la linea 42 per vederne gli effetti!).

```
10 'ragfor: "for" per demo
20 CLS:KEY OFF:INPUT "Risol.=",Z
25 SCREEN Z
30 INPUT;" r=",R         '(2;5;25)
40 INPUT;"    y=x:",S   '(1;2)
45 IF S=0 THEN 40
50 CLS
60 FOR X=0 TO 250 STEP R
70 LINE(X,X/S)-(639-X,399-X/S),,B
80 NEXT
90 INPUT"",A
100 GOTO 20
```

```
10 'ragfor1:      "for" per demo
20 CLS:KEY OFF:INPUT "Risol.=",Z
22 IF Z=0 THEN 20
25 SCREEN Z
30 INPUT;" r=",R         '(2;5;25)
40 INPUT;"    y=x:",S   '(1;2)
41 IF S=0 THEN 40
42 WINDOW (0,0)-(639,399)
45 VIEW (150,1)-(318,197),,1
50 CLS
60 FOR X=0 TO 250 STEP R
70 LINE(X,X/S)-(639-X,399-X/S),,B
80 NEXT
90 INPUT"",A
100 GOTO 20
```

In ragfor2 e' possibile replicare il disegno delle cornici in una seconda finestra (linea 82): questa possibilita' non viene offerta (il filtro e' alla linea 81) se la risoluzione scelta e' quella piu' bassa (screen 1), perche' in tal caso la view andrebbe fuori del campo di definizione massimo consentito dalla screen 1 (319,199).

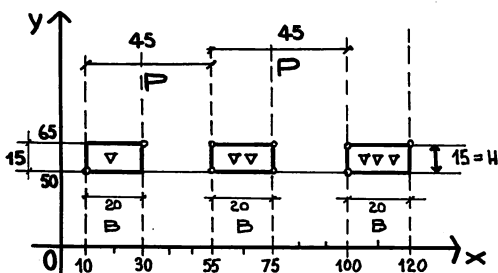
In ragfor3, infine, la gestione delle due finestre e' piu' articolata in quanto, ogni volta che, a parita' di risoluzione, si desidera cambiare parametri di cornice, questa viene generata nella prima finestra (linea 45), che parte sempre pulita, e poi riprodotta, sovrapposta alle precedenti nella seconda (linea 82).

Se si batte il carattere [/] allora si azzera la seconda finestra e in essa viene replicata la cornice presente nella prima.

Nella figura 5.6 riportiamo le copie di alcune passate dei programma ragfor nelle diverse varianti.

```
10 'ragfor2:       "for" per demo
20 CLS:KEY OFF:INPUT "Risol.=",Z
22 IF Z=0 THEN 20
25 SCREEN Z
30 INPUT;" r=",R          '(2;5;25)
40 INPUT;"    y=x:",S   '(1;2)
41 IF S=0 THEN 40
42 WINDOW (0,0)-(639,399)
45 VIEW (150,1)-(318,197),,1
50 CLS
60 FOR X=0 TO 250 STEP R
70 LINE(X,X/S)-(639-X,399-X/S),,B
80 NEXT
81 IF Z=1 THEN 90
82 VIEW (350,1)-(450,197),,1
85 FOR X=0 TO 250 STEP R
87 LINE(X,X/S)-(639-X,399-X/S),,B
88 NEXT
90 INPUT"",A
100 GOTO 20
```

```
10 'ragfor3:......"for" per demo
20 CLS:KEY OFF:INPUT "Risol.=",Z
22 IF Z=0 THEN 20
25 SCREEN Z
27 WIDTH 17
30 INPUT;" r=",R          '(2;5;25)
40 INPUT "    y=x:",S   '(1;2)
41 IF S=0 THEN 40
42 WINDOW (0,0)-(639,399)
45 VIEW (150,1)-(318,197),,1
50 CLS
60 FOR X=0 TO 250 STEP R
70 LINE(X,X/S)-(639-X,399-X/S),,B
80 NEXT
81 IF Z<2 THEN 90
82 VIEW (350,1)-(638,197),,1
85 FOR X=0 TO 250 STEP R
87 LINE(X,X/S)-(639-X,399-X/S),,B
88 NEXT
90 QQ$=INPUT $(1)
92 IF QQ$="/" THEN 50 ELSE GOTO 30
```

Terminiamo l'argomento sui cicli con un programma di generazione di rettangoli che servira' piu' avanti per tracciare i risultati sotto forma di istogrammi. Si voglia ripetere n volte un rettangolo di base B e altezza H a distanza orizzontale P l'uno dall'altro. Con riferimento alla figura 5.7, il vertice inferiore sinistro del primo rettangolo sia localizzato nel punto V1 di coordinate (10,50).

```
▽ LINE(10,50)-(10+20,50+15),,BF
▽▽ LINE(10+45,50)-(10+45+20,50+15),,BF
▽▽▽ LINE(10+2*45,50)-(10+2*45+20,50+15),,BF

FOR K=0 TO N-1
LINE(10+K*P,50)-(10+B+K*P,50+H),,BF
NEXT
```

Figura 5.7

Si tratta ora di esprimere geometricamente il punto generico V appartenente ai rettangoli successivi, sfruttando il principio dell'iterazione. Per l'enunciato del problema sara':

```
V1=(10+P,50)
V2=(10+2*P,50)
.
.
.
VN=(10+N*P,50)
```

Se si trattasse di accendere un punto sarebbe sufficiente scrivere:

```
FOR K=0 TO N-1
PSET (10+K*P,50)
NEXT
```

Ma l'obiettivo e' quello di disegnare un rettangolo e allora il corpo della FOR...NEXT sara' costituito da una **line** cosi' articolata:

```
LINE(10+K*P,50)-(10+B+K*P,50+H)
```

Nella figura 5.7.sono anche riportate le tre **line** ottenute sostituendo i valori geometrici per K=0,1,2.
Il programma, registrato col nome **retcom**, e' il seguente:

```
10 'retcom: composizione di rettangoli
20 CLS:KEY OFF:INPUT "Risoluz.=",Z
25 SCREEN Z
60 INPUT;"n. ",N
70 INPUT;"   rettangoli (H x B ) = ",H
80 INPUT;" x ",B
90 INPUT;"                        a distanza -> ",P
100 FOR K=0 TO N-1
110 LINE(10+K*P,50)-(10+B+K*P,50+H),,BF
120 NEXT
130 INPUT"",W
140 GOTO 20
```

Nella figura 5.8 la copia grafica per una passata del programma retcom; nella figura 5.9 la versione retcom1 in chiave finestre.

```
10 'retcom1: composizione di rettangoli
20 CLS:KEY OFF:INPUT "Risoluz.=",Z
25 SCREEN Z:WIDTH 19:WINDOW (0,0)-(319,199)
40 VIEW (162,1)-(318,197),,1
60 INPUT;"n. ",N
70 INPUT;"   rettangoli (H x B ) = ",H
80 INPUT;" x ",B
90 INPUT;"                        a distanza -> ",P
100 FOR K=0 TO N-1
110 LINE(10+K*P,50)-(10+B+K*P,50+H),,BF
120 NEXT:INPUT"",W:GOTO 20
```

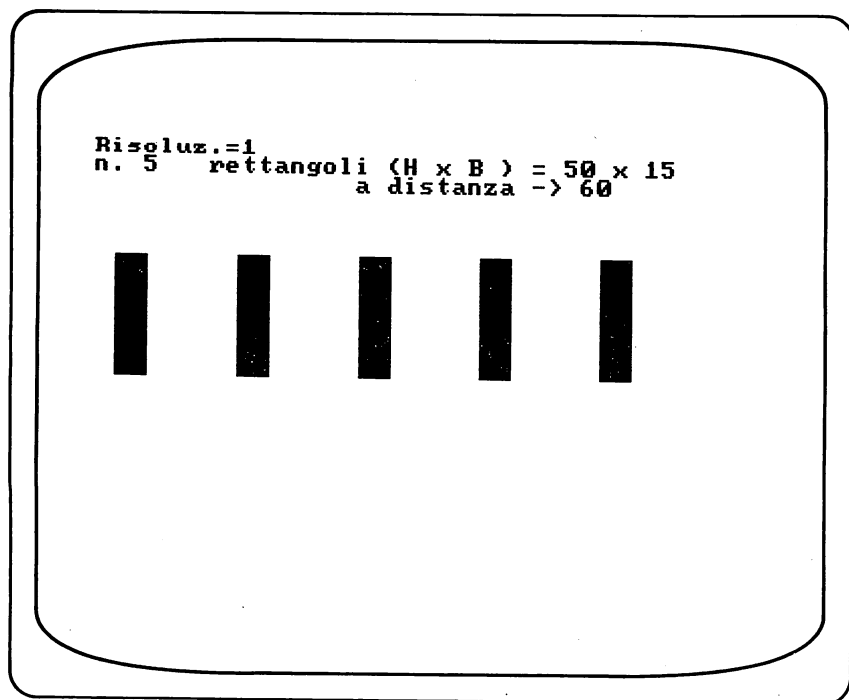



Figura 5.8

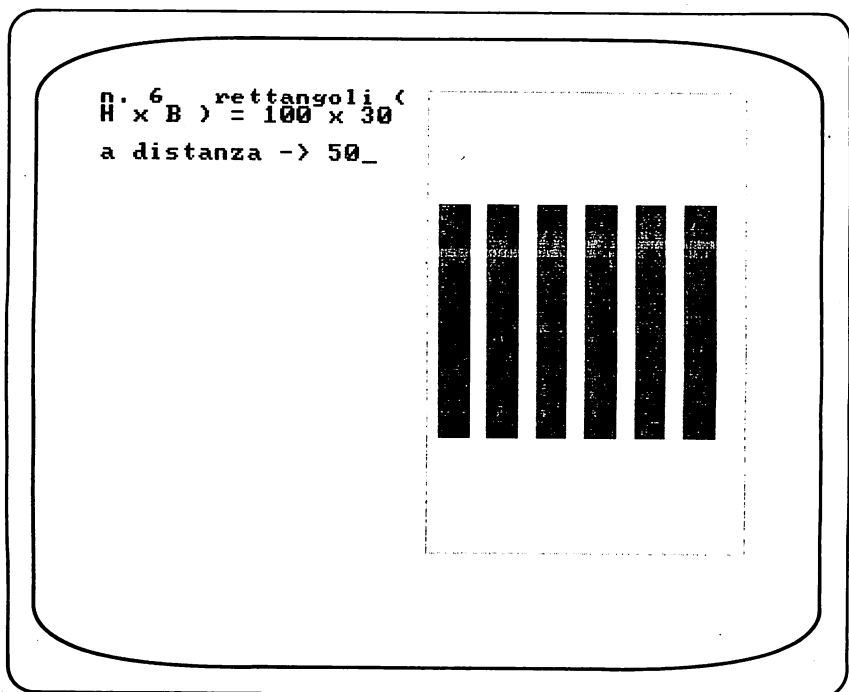



Figura 5.9

5.2. FRASI LOGICHE E DI CONTROLLO

Nei programmi che abbiamo finora elaborato si e' presentata piu' volte la necessita' di modificare l'esecuzione del programma in modo che non seguisse l'ordine stretto dei numeri di linea. Esempi di GOTO sono i piu' semplici perche' quando M24 incontra nel programma una tale istruzione salta **incondizionatamente** alla linea indicata alla destra di GOTO.

Esempio:

```
10 .........
20 INPUT.....
30 PRINT.....
40 GOTO 20...
50 .........
```

La GOTO alla linea 40 ordina alla macchina di saltare alla linea 20 anziche' eseguire la linea immediatamente successiva di numero piu' elevato. Si puo' tuttavia, in alcuni casi, desiderare di far saltare il programma a una linea fuori sequenza, ma su condizione.

Forniamo alcuni esempi di tali istruzioni cosiddette di salto condizionato, insieme alla sintassi di alcune della piu' complete espressioni di controllo:

a) IF [espressione] GOTO [numero di linea] ELSE [frase ELSE]
b) IF [espressione] THEN [frase THEN]
c) IF [espressione] THEN [frase THEN] ELSE [frase ELSE]

Un esempio del caso c):

```
IF    A > B    THEN   C = A-B    ELSE   D = B-A
```

L'espressione dopo le parole riservate IF, THEN, ELSE puo' essere aritmetica (a=b+c), di relazione (A>B) oppure logica (A>B AND B>C). Nell'ultimo caso l'ordine di priorita' di esecuzione degli operatori logici e': NOT, AND, OR, XOR, IMP, EQV.

Le espressioni di confronto, come A>B+C, hanno un risultato limitato ai valori 0 oppure 1; il risultato zero viene convenzionalmente assunto quando l'espressione di confronto e' falsa, cioe' quando A e' minore di B+C. Segue la tabella delle verita' che descrive i risultati delle operazioni logiche tra gli operatori X e Y:

X	NOT	X	Y	X AND Y	X OR Y	X XOR Y	X IMP Y	X EQV Y
0	1	0	0	0	0	0	1	1
1	0	0	1	0	1	1	0	1
		1	0	0	1	1	0	0
		1	1	1	1	0	1	1

Le [frasi THEN] e le [frasi ELSE], ossia cio' che scriviamo dopo le parole THEN (allora) e ELSE (altrimenti) possono essere molto complesse, sia perche' dopo ciascuna di esse possiamo concatenare altre IF, sia perche' possono contenere istruzioni Basic o il numero della linea a cui saltare.

Il programma logi1 seguente fornisce un esempio di operazioni logiche concatenate a una stessa linea di programma, nella quale l'unico limite e' la sua lunghezza fisica che non puo' superare la capacita' del buffer di tastiera, cioe' 255 caratteri.

```
10 'logi1: esempi di logiche e salti su condizione
20 CLS
30 INPUT; "a= ",A
40 INPUT; "  b= ",B
50 INPUT"   c= ".C
60 IF A=10 THEN IF B=20 THEN IF C=30 THEN PRINT "A+B+C= "A+B+C
   ELSE IF (C>0) AND (B>C) THEN PRINT "(B-C)/C= "(B-C)/C ELSE
   PRINT " 2x(B^3-C)= 2*(B^3-C): PRINT "fine caso particolare":
   PRINT: GOTO 30
70 PRINT: print"fine"
80 GOTO 30
```

La linea 60, in parole correnti, si potrebbe tradurre cosi': **se** le condizioni A=10, B=20 e C=30 sono tutte e tre contemporaneamente soddisfatte **allora** stampa A+B+C, **altrimenti** -basta che almeno una non sia soddisfatta (in questo caso conta l'ultima scritta, C=30)- **se** entrambe le condizioni C>0 e B>C sono vere, **allora** calcola e stampa (B-C)/C; **altrimenti** calcola e stampa il doppio del cubo di B meno C. Stampa "fine caso particolare" e torna alla linea 30.

```
RUN
a= 10  b= 20  c= 30
A+B+C=  60

fine
a= 10  b= 20  c= 0
2x(b^3-c)=  16000
fine caso particolare

a= 9  b= 20  c= 30

fine
a= 10  b= 21  c= 30

fine
a= 10  b= 20  c= 1
(B-C)/C=  19

fine
a= 10  b= 20  c= 22
2x(b^3-c)=  15956
fine caso particolare

a= 10  b= 20  c= 19.999
(B-C)/C=  4.997503E-05

fine
a=
```

Gli unici casi improduttivi sono quelli in cui le condizioni a=10 e b=20 non sono soddisfatte: infatti una catena di IF...THEN-...IF...THEN puo' venir interrotta con una ELSE, ma la [frase ELSE] che la segue puo' avere effetto solo se le condizioni precedenti poste dalla IF a monte sono soddisfatte.

Nello scrivere le espressioni logiche va tenuto presente che il risultato su cui operano e' del tipo VERO oppure FALSO (1 oppure 0). Ad esempio, l'istruzione:

```
70 IF A>B AND B=<C THEN 100
```

fa saltare all'istruzione 100 se la condizione (A>B AND B=<C) e' vera (cioe' A maggiore di B e, **insieme**, B uguale o minore di C). Se la condizione e' falsa, cioe' A minore di B oppure B maggiore di C) verra' eseguita l'istruzione successiva.

Inoltre si tenga presente che l'ordine di priorita' degli operatori logici puo' essere modificato, alla maniera di quelli numerici, con l'uso delle parentesi. Ad esempio l'ordine di priorita' di:

```
(A<B AND (C<D OR E>F))
```

non e' quello di

```
NOT A<B AND C<D OR E>F.
```

5.3 FRASI PER IL CONTROLLO DEI SOTTOPROGRAMMI

Nel progettare un programma si puo' decidere di usare piu' volte un gruppo di istruzioni. Esempi di questo tipo sono le intestazioni fisse per l'edizione di particolari stampati, per le routine grafiche (istogrammi, tabelle, diagrammi...), matematiche, etc. In tali situazioni non e' necessario **riprodurre** quel gruppo di istruzioni tutte le volte che servono, se e' possibile isolarle, in blocco, rendendole fisicamente e logicamente indipendenti. Esiste, infatti, in Basic, un meccanismo di due istruzioni (**GOSUB...RETURN**) che, quando si ha bisogno di quel gruppo (subroutine), basta andare (**GO**) a quella **SUB**routine abbandonando una certa linea di programma e poi, al termine, tornare (**RETURN**) nel punto da cui si era staccato. Il programma **somma**, riportato in figura 5.9, da' un esempio di due subroutine: quella alla linea 90 (somma dei numeri) e quella alla linea 200 (somma dei quadrati). Dal diagramma di flusso affiancato si puo' notare che le due subroutine vengono **chiamate** con istruzioni del tipo GOSUB [numero di linea]. Le subroutine si concludono entrambe con la frase fissa RETURN, che, come prima dicevamo, fa tornare all'esecuzione del programma principale. Il rientro avviene, piu' precisamente, al numero di linea immediatamente successivo a quello in cui e' scritta la GOSUB di partenza.

Nel programma e' utile seguire il movimento delle scelte dell'operatore e il conseguente ruolo della GOTO alle 70 e 180. Queste, infatti, essendo le linee successive alle GOSUB, sono anche quelle di rientro dopo l'esecuzione dei sottoprogrammi. Naturalmente, per ottenere gli effetti e la dinamica del programma **somma**, le strade sono infinite. Come le migliorie che potete apportare. Notate, comunque, l'attesa programmata alla linea 150 (ricordate il programma forat al paragrafo 5.1 ?): non ha alcuna utilita' logica -e quindi puo' essere soppressa senza far danni al programma- ma e' un piccolo

```
10 'somma:somme aritmetica e geometrica di N numeri naturali
20 KEY OFF:CLS:INPUT "RISOL.=",Z
25 SCREEN Z
30 INPUT"N (intero positivo, >45) --> ",N%
40 '
50 IF N%=<0 OR N%>45 GOTO 30 ':se N>45, Q supererebbe la capa-
60 GOSUB 90                  ' cita' massima di registrazione
70 GOTO 30                   ' dei numeri interi (32767)
80 '
90 'somma dei primi N numeri naturali
100 S%=(N%^2+N%)/2
110 LOCATE 2,34
120 PRINT"* somma aritmetica da 1 a"N%"= "S%
130 '
140 PRINT:PRINT:LOCATE 15,1
150 FOR T=1 TO 5000 :NEXT         'attesa di circa un secondo
160 INPUT" per calcolo somma geom. premere 1       poi  <CR> --> ",G
165 IF G=0 THEN 140
170 IF G=1 GOTO 195
180 GOTO 270
190 '
195 GOSUB 200
200 'somma dei quadrati dei primi N numeri
210 Q%=(2*N%^2+3*N%+1)*N%/6
220 LOCATE 3,34
230 PRINT"* somma geometrica da 1 a"N%"="Q%
240 '
250 INPUT"",W
260 RETURN
270 RETURN

Ok
RUN
RISOL.=3
N (intero positivo, >45) --> 25
* somma aritmetica da 1 a 25 =  325


 per calcolo somma geom. premere 1       poi  <CR> --> 1
* somma geometrica da 1 a 25 = 5525

* somma geometrica da 1 a 25 = 5525

N (intero positivo, >45) --> 45
* somma aritmetica da 1 a 45 =  1035


 per calcolo somma geom. premere 1       poi  <CR> --> 1
* somma geometrica da 1 a 45 = 31395

* somma geometrica da 1 a 45 = 31395

N (intero positivo, >45) --> 46
N (intero positivo, >45) --> 20
* somma aritmetica da 1 a 20 =  210
```

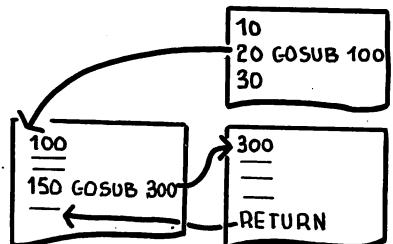
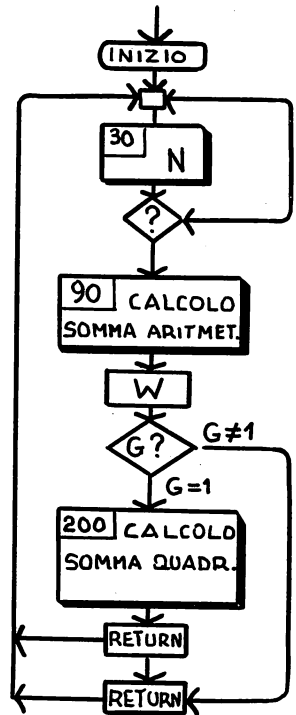



Figura 5.10

particolare di stile: serve, infatti, a separare temporalmente la visualizzazione del primo risultato con quella della domanda [per calcolo somma geometrica, premere 1....] (vedi la figura 5.11).

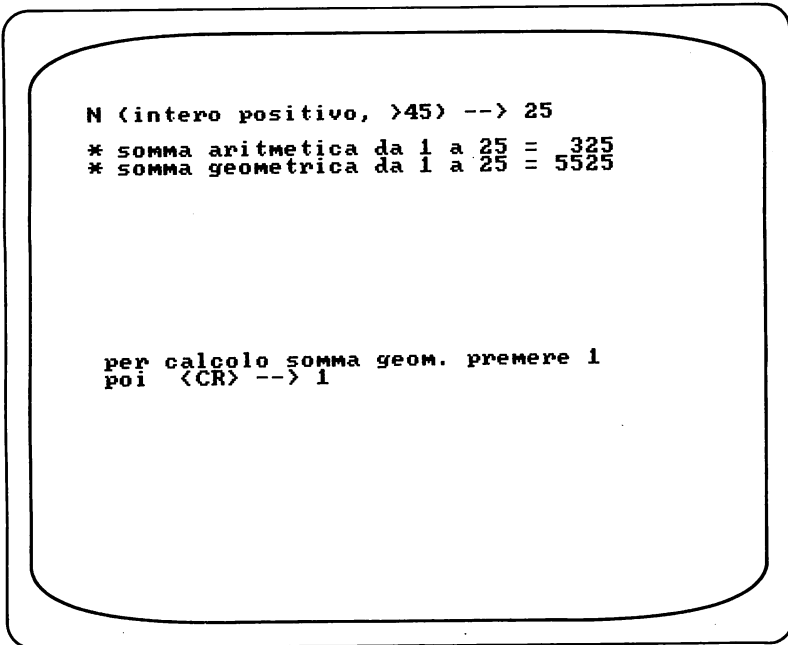



Figura 5.11

Altri particolari di stile riguardano il controllo della posizione del video, attraverso le LOCATE delle linee 110 e 220, in accoppiata con le PRINT alle 120 e 230. La sintassi delle GOSUB...RETURN e' illustrata nella figura 5.12.

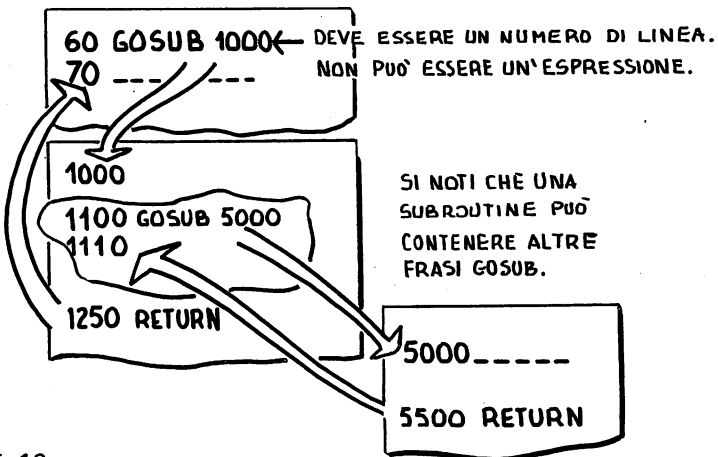



Figura 5.12

L'istruzione GOSUB provoca, dunque la deviazione dall'ordine sequenziale del programma e il salto alla linea in cui ha inizio il sottoprogramma.

Se il programma e' scritto correttamente ogni subroutine deve avere alla fine delle proprie istruzioni una linea che contiene il verbo RETURN.

Con tale istruzione, nell'ambito dello stesso sottoprogramma, si comanda alla macchina di riprendere il controllo del programma principale.

Da quanto detto appare chiaro che la subroutine ci evita di riscrivere parti di programma da ripetere piu' volte, ma ci puo' anche permettere di utilizzare sottoprogrammi prodotti in altra sede.

L'utilita' di aver progettato un programma strutturandolo in sottoprogrammi e' che quel gruppo di istruzioni di cui quel sottoprogramma e' composto, puo' essere programmato da persone diverse e in momenti diversi.

Inoltre, se il sottoprogramma e' ben generalizzato (vedi un programma di ordinamento, una composizione grafica ricorrente, etc.), puo' essere riutilizzato in altri programmi.

Addirittura la progettazione di nuovi programmi puo' essere condizionata dall'esistenza di certi **mattoni** preconfezionati che possono essere immediatamente riutilizzati, salvo pochi adattamenti.

Della tecnica dei concatenamenti parleremo in tema di fusione (Merge) di programmi al capitolo 9: qui conviene sottolineare il principio che sta sotto e che riguarda non solo la struttura logica di un programma ma il modo di costruirlo.

A partire dall'obiettivo finale, scendendo verso il basso si procede alla scomposizione del problema in tanti problemi minori, ciascuno dei quali costituisce un obiettivo intermedio.

Tale modo di risolvere un problema porta alla progettazione di un programma a struttura aperta, in cui e' possibile apportare modifiche ad alcune sue parti senza comprometterne la validita' generale. La corrispondente programmazione viene quindi chiamata **strutturata** perche' consente di schematizzare efficacemente un processo logico.

La componente grafica come alternativa di intepretazione dei risultati aggiunge infine un ulteriore motivo per organizzarsi una scenografia intercambiabile fino all'ultimo momento.

E questo puo' ottenersi solo se i diversi "strati" del programma sono suscettibili di un certo grado di concatenazione.

Questo modo di far software, certamente piu' semplice di quello tradizionale che produceva programmi compatti e difficilmente modificabili, consente perfino di concepire un programma con diversi livelli di difficolta', in modo che l'utilizzatore vi si possa gradualmente adattare.

Queste sono solo alcune riflessioni di carattere generale. In pratica avere un programma suddiviso in tanti moduli sufficientemente piccoli e che possono essere singolarmente collaudati, semplifica molto la correzione del programma.

Un'altra considerazione riguarda l'occupazione di memoria. Infatti una programmazione a moduli consente l'impiego di un programma di grandi dimensioni anche su macchine con capacita' di memoria relativamente modeste. L'argomento va naturalmente trattato tenendo conto anche degli aspetti prestazionali.

5.4 COME CAMBIARE SCALA

Abbiamo gia' visto che nel repertorio Gwbasic di M24 c'e' un'istruzione, la **window**, che consente di fissare le coordinate dei vertici dello schermo grafico: in tal modo le dimensioni delle figure rappresentate sul video possono essere modificate senza ridisegnarle, attraverso la semplice riassegnazione delle suddette coordinate. Proviamo, per un attimo e a puro titolo educativo, come potremmo cavarcela senza questa potentissima istruzione. Il programma **scal314** e' stato preparato come esempio didattico per illustrare modalita' e relazioni per compiere delle trasformazioni di alcune funzioni grafiche elementari come il disegno di un rettangolo mediante l'istruzione **line** (,)-(,),,B.

```
10 'scal314:trasl. e trasformazione di rettangolo
20 SCREEN 3:CLS:KEY OFF
43 INPUT"b --> ",B
44 INPUT"h --> ",H
50 INPUT "fattore di scala --> ",S
60 INPUT "fattore di trasl.--> ",T
65 INPUT "Xo=",X
67 INPUT "Yo=",Y
70 GOSUB 1070
110 LINE(X1,Y1)-(X2,Y2),,B
120 QQ$=INPUT$(1)
125 IF QQ$="/" THEN 50
130 GOTO 65
1070 X1=S*X+T
1080 Y1=S*Y+T
1090 X2=X1+S*B
1100 Y2=Y1+S*H
1200 RETURN
```

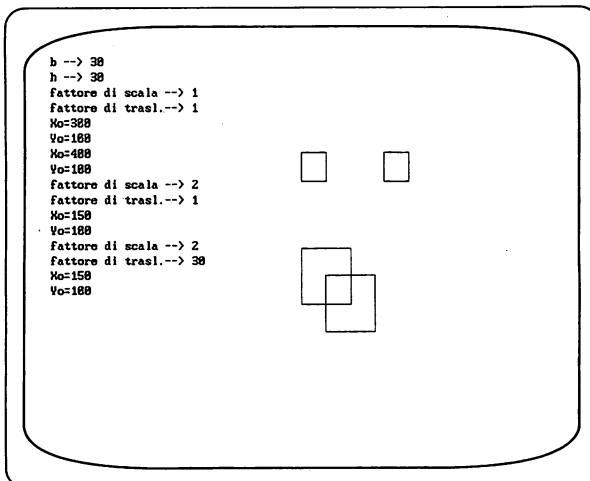



Figura 5.13

Il programma e' strutturato in modo che prima di eseguire l'istruzione grafica i valori delle coordinate dei vertici del rettangolo (X1,Y1; X2,Y2) vengano normalizzati da un sottoprogramma (linee 1070÷1200) che contiene alcune istruzioni di assegnazione in funzione dei parametri **s** e **t**, fattori di scala e di traslazione.
Come appare dalla figura 5.13 sembra che anche il fattore scala faccia traslare la figura: in effetti, per valori di scala maggiori di 1, e' la misura della sua distanza dall'origine che si dilata come ogni punto del piano. Ma torniamo all'istruzione **window** che appare cosi' strutturata:

```
window (Xmin,Ymin)-(Xmax,Ymax)
```

In essa vengono dichiarate le coordinate dei vertici opposti della finestra attiva in quel momento (vedi figura 5.14) riferiti all'origine del sistema cartesiano nel quale sono state definite le grafiche che intendiamo riportare in scala. Tale sistema viene comunemente chiamato **sistema** di riferimento **utente**.

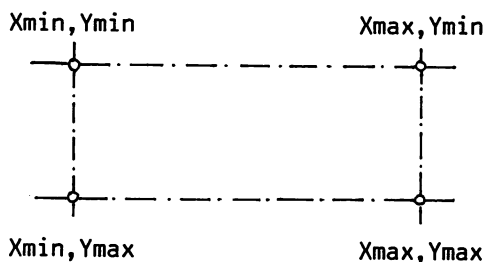



Figura 5.14

Un esempio di quanto appena affermato si puo' avere riscrivendo il programma gabset, che ridenominiamo **triwind**.

```
10 'triwind:finestre in scala
20 CLS:KEY OFF:SCREEN 1
24 VIEW SCREEN (153,40)-(273,85),,1
90 PSET (120,85)
100 LINE (79,149)-(237,149)
110 LINE -(159,49)
120 LINE -(79,149)
140 QQ$=INPUT$(1)
150 GOTO 20
```

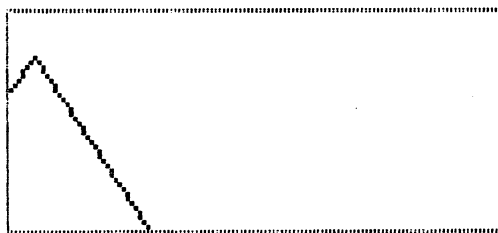



Figura 5.15

In tale programma, anziche' disegnare un punto, si disegna un triangolo, ma di questo si riesce ad ottenere tracce di generazione solo introducendo l'opzione screen nella view alla linea 24 che rende nella finestra-tassello la parte alta della figura (figura 5.15). Senza tale opzione poiche' le coordinate dei vertici del triangolo sono esterne alla finestra non vedremmo alcuna traccia. Infatti la finestra aperta dalla view puo' rappresentare punti di coordinate compatibili con lo spazio a disposizione, che e' quello racchiuso dal rettangolo compreso dai vertici opposti (espressi in valore reale): (0,0); (120,85). In tale area non sono pertanto rappresentabili i vertici del triangolo programmato alle linee 100, 110, 120 (figura 5.16).

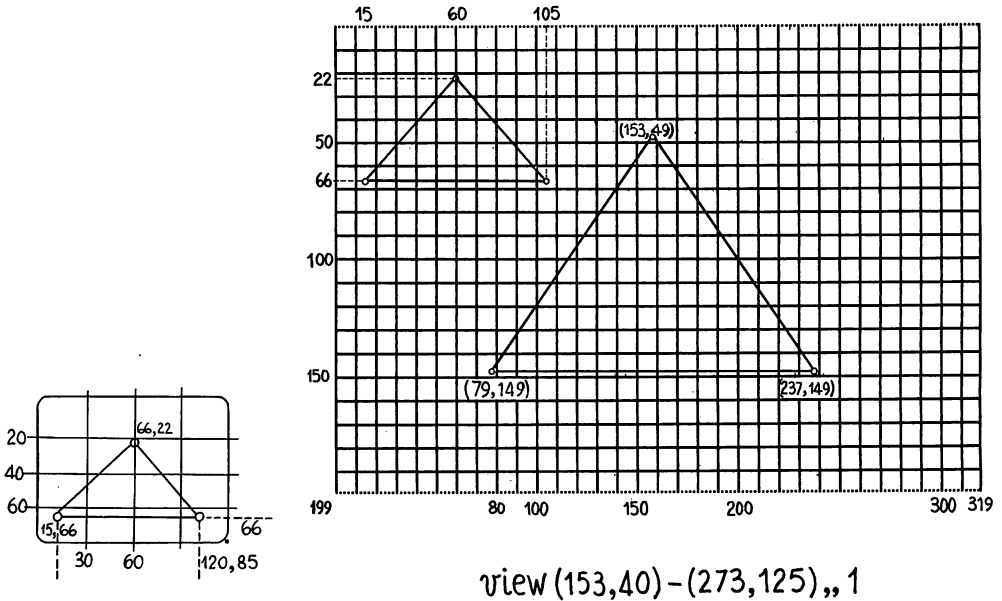



Figura 5.16

A riprova di cio' aggiungiamo alle linee 85, 90, 95 un altro triangolo i cui vertici siano compatibili con lo spazio racchiuso dalla finestra. Otteniamo quindi il programma **triwind1**.

```
10 'triwind1:finestre in scala
20 CLS:KEY OFF:SCREEN 1
24 VIEW (153,40)-(273,125),,1
85 LINE (15,66)-(105,66)
90 LINE -(60,22)
95 LINE -(15,66)
100 LINE (79,149)-(237,149)
110 LINE -(159,49)
120 LINE -(79,149)
140 QQ$=INPUT$(1)
150 GOTO 20
```

```
10 'triwind2:finestre in scala
20 CLS:KEY OFF:SCREEN 1
22 WIDTH 19
24 VIEW (153,40)-(273,125),,1
82 WINDOW (0,0)-(319,199)
85 LINE (15,66)-(105,66)
90 LINE -(60,22)
95 LINE -(15,66)
100 LINE (79,149)-(237,149)
110 LINE -(159,49)
120 LINE -(79,149)
140 QQ$=INPUT$(1)
150 GOTO 20
```

Aggiungendo, infine, un'istruzione window (0,0)-(319,199), prima della view, (programma **triwind2**) otteniamo una vera finestra proiettiva (figura 5.17) con le convenzioni cartesiane. Per mantenere la convenzione primitiva (origine in alto a sinistra e asse delle ordinate verso il basso) basta scrivere la parola screen dopo la window, cioe': window screen (0,0)-(319,199). L'effetto e' visibile nalla figura 5.18.

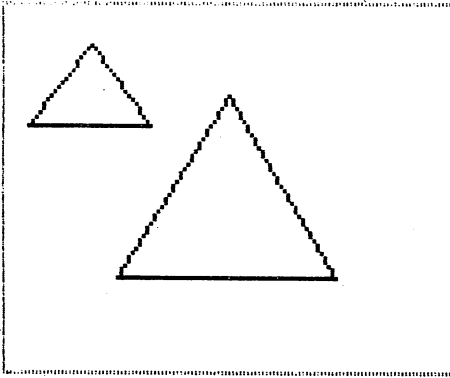



Figura 5.17

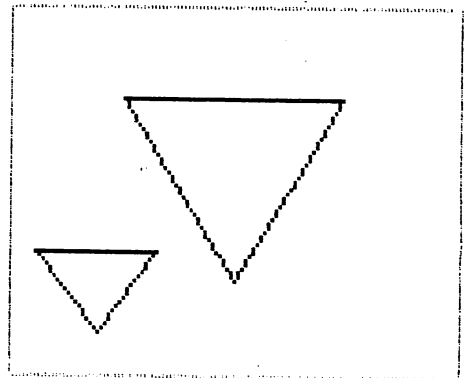



Figura 5.18

Cio' che segue ora vale per la massima risoluzione grafica operata dal comando **screen 3**, ma, ovviamente, si puo' ripetere il ragionamento anche per le altre (screen 1 e 2). Cosi', dopo aver disegnato con la line (0,0)-(639,399) il massimo rettangolo rappresentabile sul video, e' possibile ridurlo in scala 1:8 se la line sara' preceduta da una window (-2240,-1400)-(2879,1799).

Infatti, considerando il rapporto dei lati del suddetto rettangolo massimo rappresentabile (come bordo estremo del video fisico) con gli analoghi del rettangolo che otterremmo una volta eseguita la riduzione in scala (figura 5.19) dovra' essere:

$$\frac{\bar{a}}{A} = \frac{\bar{b}}{B} = \frac{1}{8}$$

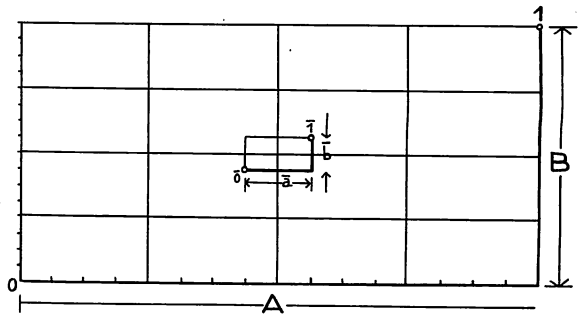



Figura 5.19

Dividiamo i lati a e b rispettivamente in 640 e 400 parti e rapportiamoli ad A e B secondo la scala assegnata (1:8); si ha cosi':

$A = 8\bar{a} = 8.640 = 5120$

$B = 8\bar{b} = 8.400 = 3200$

Posizionando l'origine degli assi $\bar{O}$ come in figura 5.19 in modo che il centro sia coincidente con quello della finestra madre le coordinate (x_o,y_o) di 0 rispetto alla nuova origine 0, saranno:

$\bar{x}_o = -(7/16)A = -(7/16)5120 = -2240$

$\bar{y}_o = -(7/16)B = -(7/16)3200 = -1400$

Inoltre le coordinate di $1(\bar{x}_1,\bar{y}_1)$, sempre riferite alla nuova origine saranno:

$\bar{x}_1 = (9/16)A = (9/16)5120 = 2880$

$\bar{y}_1 = (9/16)B = (9/16)3200 = 1800$

Riduzioni minori (ad esempio 1:4 e 1:2) sono realizzabili con window cosi' articolate:

1:4 WINDOW (-960,-600)-(1599,999)

1:2 WINDOW (-320,-200)-(959,599)

Queste riduzioni sono avvenute nel rispetto delle proporzioni tra i lati, ma, volendo, avremmo potuto alterare anche il loro rapporto. Per valutare queste possibilita', analizziamo il seguente programma wincart9.

```
10 'wincart9:finestre in scala
20 CLS:KEY OFF:SCREEN 3
30 INPUT"xmin = ",XMIN
40 INPUT"ymin = ",YMIN
50 INPUT"XMAX = ",XMAX
60 INPUT"YMAX = ",YMAX
70 IF XMIN=XMAX OR YMIN=YMAX THEN 30
80 WINDOW (XMIN,YMIN)-(XMAX,YMAX)
90 LINE (0,0)-(639,399),,B
100 LINE (159,99)-(479,99)
110 LINE -(319,299)
120 LINE -(159,99)
140 QQ$=INPUT$(1)
150 GOTO 30
```

Nel programma l'istruzione window e' stata resa parametrica, come e' possibile fare per la maggior parte delle istruzioni Basic.
 Limitiamoci per il momento a inserire nell'argomento della window quattro semplici variabili manovrabili dall'esterno. Per valutare l'effetto della riduzione abbiamo disegnato con la 90 la massima cornice e, nel suo interno, il solito triangolo equilatero con le istruzioni alle linee 100, 110 e 120. Sostituendo i valori sopraindicati per le riduzioni 1:2, 1:4 e 1:8 alle Xmin, Ymin, Xmax, Ymax nelle INPUT di questo programma si possono verificare sperimentalmente gli effetti di scala (figura 5.20). Come si puo' notare, ogni figura e' centrata nel video e non ha subito alcuna deformazione, ne' traslazione rispetto ai propri riferimenti.

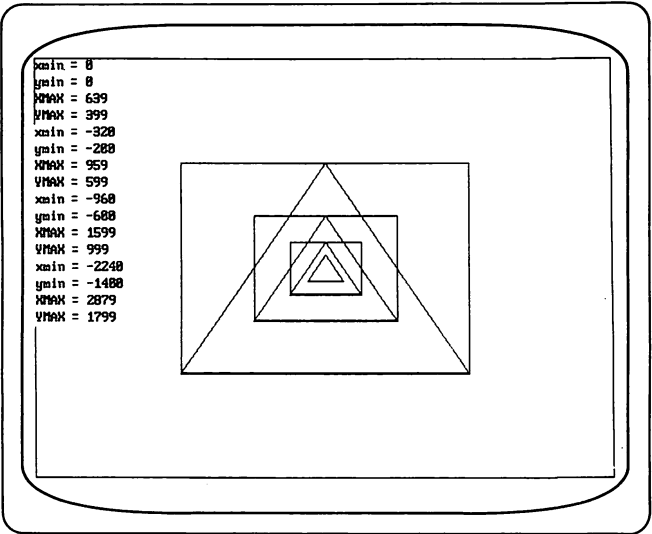



Figura 5.20

La seguente tabella fornisce per fattori di scala in progressione geometrica, da 1:1 a 1:32, i corrispondenti parametri della window.

scala	Xmin	Ymin	Xmax	Ymax
1:1	0	0	639	399
1:2	-320	-200	959	599
1:4	-960	-600	1599	999
1:8	-2240	-1400	2879	1799
1:16	-4800	-3000	-5439	3399
1:32	-9920	-6200	10559	6599
1:1	-320	-200	320	200
1:2	-640	-400	640	400
1:4	-1280	-800	1280	800
1:8	-2560	-1600	2560	1600
1:16	-5120	-3200	5120	3200

Notiamo che la differenza in valore assoluto tra le ascisse e le ordinate cresce con la stessa ragione. Infatti se X1-X0=639 e Y1-Y0=399 sono le suddette differenze per la scala 1:1, per quella 1:2 tali differenze raddoppiano e cosi' via. Nelle ultime 5 righe della tabella abbiamo anche riportato 5 fattori di scala a cui corrispondono parametri che pur rispettando la legge sulle differenze prima esaminate, hanno tutte la proprieta' di avere in comune il vertice basso sinistro della cornice (figura 5.21).

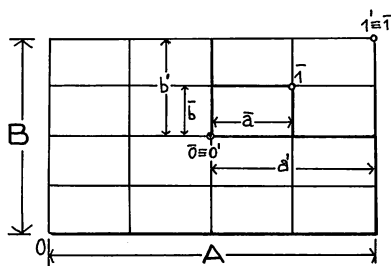



Figura 5.21

In questo caso particolare, ripetendo lo stesso ragionamento che ci ha condotto ai valori per la scala 1:8 quando le cornici dovevano tutte avere in comune il centro, ora, ad esempio per la scala 1:4, i parametri si ottengono esprimendo il numero delle parti in cui i lati sono suddivisi nel rapporto:

$$\frac{a}{A} = \frac{b}{B} = \frac{1}{4} \quad ; \qquad \text{da cui} \qquad A = 4a = 4.640 \qquad B = 4b = 4.400$$

La vecchia origine 0 rispetto al centro 0' dello schermo (che e' ora la nuova origine) avra' allora coordinate:

$$x'_0 = -(4/8)A = -A/2 = -(4.640)/2 = -1280$$
$$y'_0 = -(4/8)B = -B/2 = -(4.400)/2 = -800$$

Analogamente il vertice (1) avra' coordinate:

$$x'_1 = (4/8)A = A/2 = 1280$$
$$y'_1 = (4/8)B = B/2 = 800$$

Per la scala 1:2 le relazioni di partenza tra i lati sono:

$$\frac{a''}{A} = \frac{b''}{B} = \frac{1}{2} \quad ; \qquad \text{da cui} \qquad A = 2a'' \qquad B = 2b''$$

$$x''_0 = -(4/8)A = -(2.640)/2 = -640$$
$$y''_0 = -(4/8)B = -(2.400)/2 = -400$$

Le coordinate di (1) rispetto alla nuova origine O' sono:

$$x''_1 = (4/8)A = 640$$
$$y''_1 = (4/8)B = 400$$

In particolare, immaginando di aver diviso il video nei quattro quadranti cartesiani con origine nel centro del video, la window (-320,-200)-(320,200) sfrutta solo il primo quadrante, (vedi figura 5.22) e cosi' via.

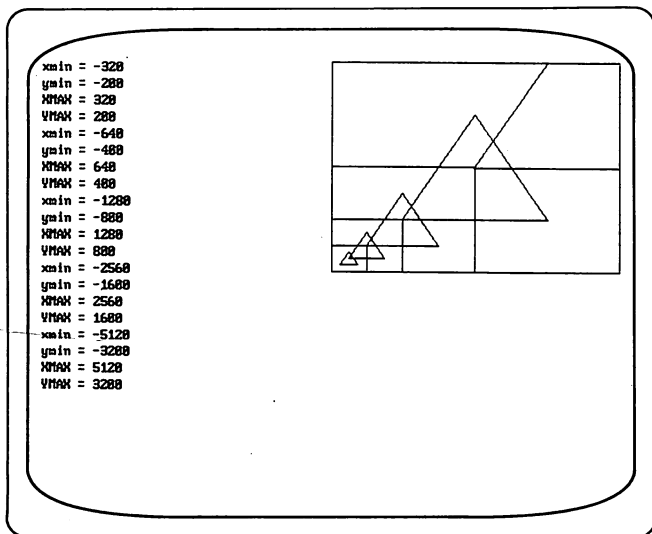



Figura 5.22

Vediamo ora che cosa accade a muovere un parametro per esempio Y1, lasciando fermi tutti gli altri. E' quello che la variante **scalvar9** del programma precedente consente di fare.

```
10 'scalvar9:finestre in scala
20 CLS:KEY OFF:SCREEN 3
25 WIDTH 19
30 INPUT"xmin = ",XMIN
40 INPUT"ymin = ",YMIN
50 INPUT"XMAX = ",XMAX
60 INPUT"YMAX = ",YMAX
70 IF XMIN=XMAX OR YMIN=YMAX THEN 30
80 WINDOW (XMIN,YMIN)-(XMAX,YMAX)
90 LINE (0,0)-(639,399),,B
100 LINE (159,99)-(479,99)
110 LINE -(319,299)
120 LINE -(159,99)
140 QQ$=INPUT$(1)
150 GOTO 60
```

Infatti, nell'ultima linea -dopo l'input sospensiva che consente di osservare gli effetti del programma per quanto tempo vogliamo- anziche' rinviare alla testa del programma, la GOTO fa saltare alla linea 60 dove troviamo la INPUT che ci consente di riciclare con altri valori di Ymax, lasciando immutati i valori degli altri 3 parametri della Window.

In questi esperimenti notiamo innanzitutto che le grafiche oltre a spostarsi verticalmente mantenendosi in una specie di corridoio centrale, si deformano notevolmente, (vedi figura 5.23).

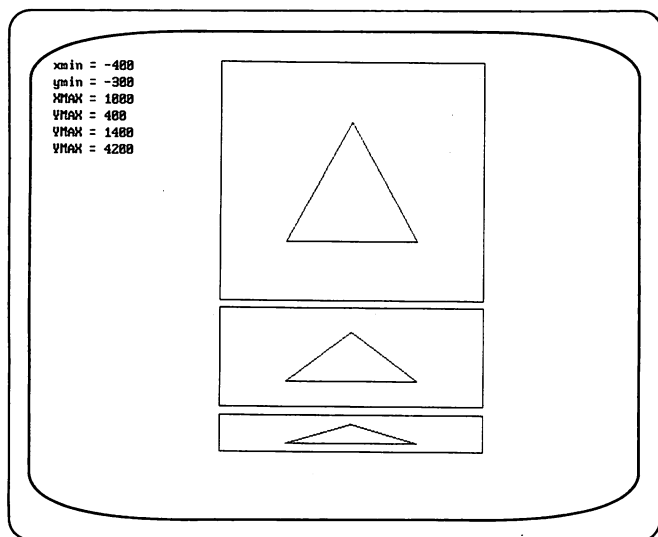



Figura 5.23

Con questo tipo di esperimenti e qualche nozione di geometria analitica elementare siamo cosi' giunti a formulare un algoritmo che controlli i diversi modi di spostare e trattare una figura sul video. E' nato cosi il programma seguente registrato col nome di **scalcom2**.

```
10 'scalcom2
20 CLS:KEY OFF
30 INPUT"Risol.= ",Z
40 SCREEN Z
50 INPUT; "scala 1:",S
60 INPUT "      +/-Trasf.=",T
70 IF T=0 THEN 60
80 K=319*S-319
90 WINDOW (-K+T,T-K)-(639+K+T,T+399+K)
100 LINE (0,0)-(639,399),,B
110 CIRCLE (100,100),30
120 QQ$=INPUT$(1)
130 GOTO 50
```

In esso i parametri esterni non sono piu' quelli della window, ma parametri globali piu' semplici da gestire e da collegare all'effetto che si desidera ottenere. Questi parametri sono:

- fattore di scala
- fattore di trasferimento

Il disegno da riprodurre e' la cornice con un cerchio. Le copie riprodotte nelle figure 5.24 e 5.25 sono state ottenute con diversi valori di scala.
Il successivo programma **scalacom** introduce un ulteriore parametro, il fattore deformante.

```
10 'scalacom
15 SCREEN 0,0,0
20 CLS:KEY OFF
30 INPUT"Risol.= ",Z
35 IF Z=0 THEN 30
40 SCREEN Z
45 WIDTH 19
50 INPUT "scala 1:",S
55 IF S=0 THEN 50
60 INPUT;"      +/-Trasf.=",T
65 LOCATE 4,1
70 INPUT "Deform=",D
75 IF D=0 THEN 70
80 K=320*S-320
90 WINDOW (-K+T,T/D-K/D)-(639+K+T,T/D+399+K/D)
100 LINE (0,0)-(639,399),,B
110 LINE (159,99)-(479,99)
115 LINE -(319,299)
118 LINE -(159,99)
120 QQ$=INPUT$(1)
125 IF QQ$="/" THEN 15
130 GOTO 70
```

La dinamica del programma prevede, come in scalvar9, la possibilita', dopo la selezione iniziale, di modificare un parametro (l'effetto deformante) a parita' degli altri due. La sospensiva alla linea 120 consente di esaminare il risultato: qualsiasi tasto battiamo sblocca l'attesa con la possibilita' di ripetere l'esperimento variando solo l'effetto deformante. Se il tasto battuto e' [/] cio' provoca il rinvio alla testa del programma.

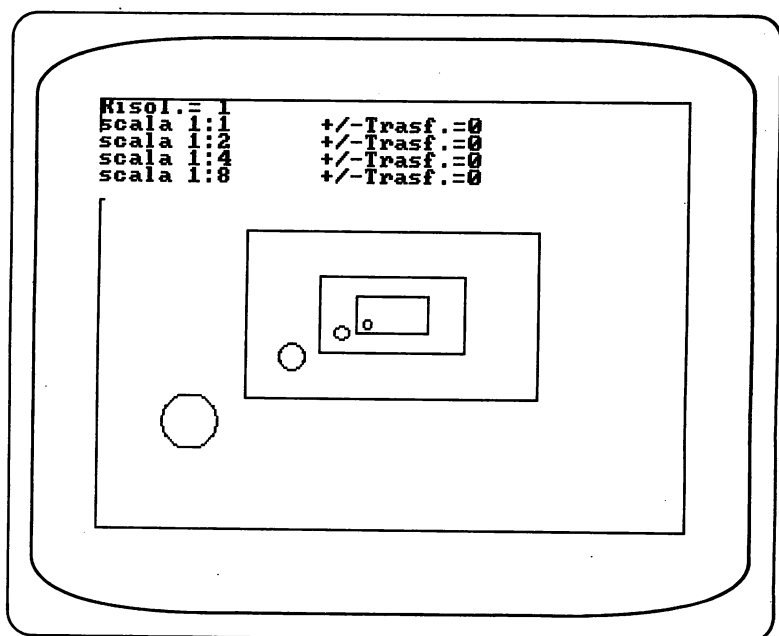



Figura 5.24

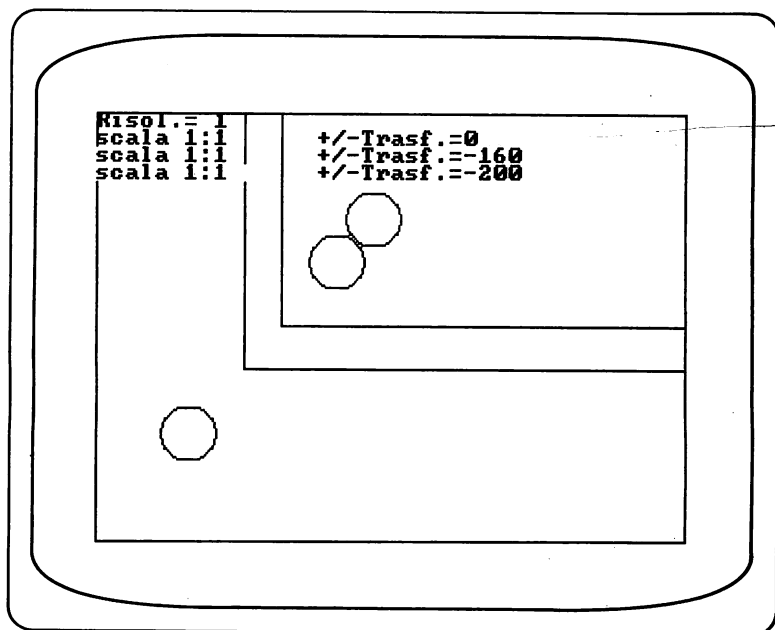



Figura 5.25

CAPITOLO
SEI

UN PO' DI MATEMATICA SU M24

In questo capitolo non intendiamo ridurre la matematica a poche situazioni pratiche, ma anzi utilizzare l'informatica per cogliere piu' facilmente alcuni concetti matematici. Vedremo la possibilita' di interpretare graficamente i risultati e allora quando riusciremo a dar corpo a certe astrazioni, incominceremo anche a perdere quel senso di diffidenza verso l'elaboratore e ad avere meno timore di perdere la purezza del pensiero matematico.

SCOPI DEL CAPITOLO

Analogamente a altri linguaggi, il Basic consente di scrivere programmi che "masticano" dati o, come si dice, elaborano dati. Nel termine **elaborare** sono compresi i diversi significati:

- calcolare, far di conto, operare sui numeri con operatori aritmetici e piu' che aritmetici;

- manipolare parole o, piu' generalmente, sequenze di caratteri (stringhe), unire due o piu' stringhe tra loro, oppure con l'operazione inversa, spezzare una stringa in piu' parti o sottostringhe;

- convertire dati da una forma di rappresentazione a un'altra, ad esempio da decimale a esadecimale (base 16).

Tratteremo tutti questi argomenti che tipicamente sanno d'informatica, ma, nello stesso tempo, sono tangenti alla matematica: sara' forse una buona occasione per riavvicinare questa materia, cosi' discussa e spesso incompresa o odiata per i suoi contenuti ritenuti astratti.

Vero e' che alcuni concetti, alla base dell'informatica, come quello di precisione e di controllo degli errori di arrotondamento, non possono essere trascurati e, tuttavia, non sono certamente tra i piu' esaltanti. Quanto accennato nel primo capitolo su questo argomento, dunque, va ora meglio meditato e opportunamente approfondito.

Altri concetti matematici indispensabili sono, come gia' osservato in altra parte del libro, quelli di variabile e di funzione. Non conviene trascurarli o male assimilarli: potrebbero provocare grosse delusioni.

L'approccio seguito e', comunque, di tipo intuitivo e non fa ricorso a eccessiva teoria. Chi desiderasse verificare l'utilita' pratica dello strumento matematico puo' estendere la lettura all'appendice A, dedicata alla costruzione di algoritmi un po' piu' elaborati, come il calcolo matriciale, il simplesso e i parametri statistici.

6.1 GENERALITA' SULLE FUNZIONI

Abbiamo tentato nel paragrafo 2.1 un primo approccio al concetto di funzione. Anche se questo libro non e' la sede per approfondirne la teoria, cio' nondimeno, poiche' continueremo a parlare di funzioni in tema di programmazione Basic dobbiamo conoscerne almeno gli aspetti essenziali.

La funzione, intesa come corrispondenza biunivoca tra gli elementi di due insiemi, e' solo un esempio particolare. Le funzioni numeriche (corrispondenza biunivoca tra insiemi di numeri) possono essere a piu' argomenti. Nell'esempio della gabbia tipografica (che immagino ricorderete) la larghezza e' una funzione a due argomenti, in quanto dipende dai valori assegnati a due variabili: DO e RP (densita' orizzontale e numero di righe per pagina).

Nello stabilire la corrispondenza tra gli insiemi degli argomenti e quello della loro funzione, in genere si formula anche la legge che elaborando i valori degli argomenti fabbrica il valore della funzione stessa. Continuando a spiegare questi concetti in forma piu' intuitiva che teorica impieghiamo ancora il tipo di rappresentazione gia' utilizzato nella figura 2.11 e applichiamola alla seguente espressione:

D=((a+b)/c)^2

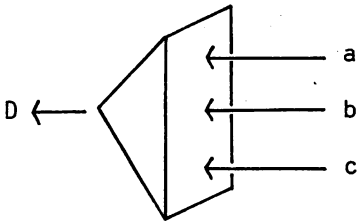
dove, evidentemente, il valore di D dipende dai valori che abbiamo assegnato ad a, b, c. Dalla tabella delle corrispondenze in figura 6.1 notiamo almeno due cose.

a	b	c	a+b	(a+b)/c	((a+b)/c)^2=D
0	0	1	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	1	2	2	4
2	0	1	2	2	4
2	1	1	3	3	9
1	2	1	3	3	9
2	2	2	4	2	4
...	...	...	...	...	...

Figura 6.1

La prima osservazione e' che nella combinazione delle tre variabili non abbiamo mai assegnato il valore zero alla c, perche' in tal caso

il valore della funzione (a+b)/c non sarebbe stato determinato (non esiste nessun numero che moltiplicato per zero dia a+b). La seconda e' che la variabile **dipendente** D puo' avere lo stesso valore per combinazioni diverse di a, b, c. In M20 le funzioni che operano tale tipo di trasformazione sulle variabili si dicono funzioni definibili a scelta dell'utente, in contrapposizione a quelle gia' definite all'interno della macchina, come le trigonometriche, le logaritmiche, etc.

Un esempio di quest'ultime e' la funzione y=x^2; in questo caso la funzione F e' l'operazione di elevazione al quadrato della variabile x: il tipo di corrispondenza che si e' stabilita tra gli insiemi dei valori di x e di y e' quello riportato nella sottostante tabella, in cui per semplicita' abbiamo considerato solo alcuni valori interi di x.

Y=x^2 ⟵	x
1	1
4	2
9	3
16	4
...	...

Ma esiste anche la corrispondenza, o funzione, inversa:

x=(y)^(-1/2) ⟵	y
1	1
2	4
3	9
4	16
...	...

In generale, dunque, scrivendo y=F(x) (si dice y e' uguale a F di x) e assegnando un valore a x si sottopone tale variabile a un processo di trasformazione che **produce** un altro valore appartenente all'insieme delle trasformate di x, che, nel nostro caso, chiameremo l'insieme dei valori di y. Esempi di funzioni gia' definite da M20 come la suddetta radice quadrata (in Basic la parola chiave e' SQR= square root) e di funzioni definibili a piacere con i relativi dettagli operativi verranno trattati nei successivi paragrafi.

6.2 LE VARIABILI NUMERICHE E LE VARIABILI STRINGA

Abbiamo fin qui considerato soltanto le variabili di tipo numerico; il Basic M20 puo', tuttavia, elaborare anche variabili non numeriche, comunemente chiamate variabili **stringa**. Il contenuto di queste ultime

puo' essere un qualsiasi carattere tra quelli generabili dalla macchina: lettere alfabetiche, simboli, numeri e combinazione di essi.

La prescrizione sulla lunghezza dei nomi sono le stesse gia' dette per le variabili numeriche. Il tipo di variabile stringa si distingue pero' dalle altre per il simbolo particolare [$] in coda al nome. Il "valore" da assegnare alla variabile stringa **deve** essere racchiuso tra virgolette. Cosi' l'istruzione

```
V$="alfa"
```

significa: assegna alla variabile stringa V$ il valore **alfa.** Quando vogliamo azzerare il contenuto di una variabile stringa bastera' impostare

```
V$=""
```

cioe' introdurre **solo** le virgolette e **niente** tra loro (neppure uno spazio). L'unico operatore numerico ammesso tra variabili stringa e' quello di addizione [+], che pero', ovviamente, ha solo significato di concatenazione di stringhe. Cosi' definite le stringhe

```
V$="alfa"    e    W$=" romeo"
```

quando impostiamo l'istruzione di stampa

```
PRINT V$+W$
```

si otterra' la fusione delle due parole stringa : alfa romeo.

Si noti che la separazione tra le due parole e' avvenuta perche' come primo carattere, in testa alla parola " romeo" abbiamo messo uno spazio. La stringa, inoltre, come valore da assegnare a una data variabile, non puo' avere piu' di 255 caratteri estratti in sequenza qualsiasi tra i codici ASCII. Questi sono anch'essi 255, e quasi a tutti corrisponde un carattere riproducibile su video (numero, lettera alfabetica o simbolo qualsiasi). La stampa di essi puo' avvenire su stampanti ad aghi o su stampanti di qualita' attrezzate con particolari margherite.

Nella figura 6.a si riportano i vari codici ASCII organizzati in una tabella da cui e' possibile ottenere i numeri d'ordine di entrata dei simboli ivi stampati.

La stringa -e' bene ripeterlo- viene considerata convenzionalmente tale, anche se contiene solo dei numeri, soltanto **se** racchiusa da virgolette "   ". Nel conteggio dei 255 caratteri le virgolette sono escluse.

Conviene, ora, riesaminare i diversi tipi di variabili **numeriche,** a seconda dei valori che vogliamo loro assegnare.

Com'e' noto il Basic puo' trattare dati numerici reali in tre modi diversi a seconda della precisione che si vuole raggiungere nei risultati: cioe' valori numerici **interi** e valori numerici **reali,** in semplice e doppia precisione. Tutti i numeri vengono introdotti in forma decimale (base 10). A seconda del modo prescelto cambia l'occupazione in memoria e la velocita' con cui il Basic elabora il dato.

INTERI E' il modo piu' efficiente di registrare i numeri, sia sotto l'aspetto occupazione di memoria, sia per la velocita' di calcolo. Come dice il nome, una variabile intera puo' contenere solo numeri interi, cioe' senza punto decimale (che nel Basic si esprime in notazione anglosassone, non con la virgola ma col punto).
Il valore di una variabile intera deve essere compreso tra -32768 e +32767. La qualifica di contenitore di numeri interi si assegna mettendo in coda al nome della variabile il simbolo %.

REALI **in semplice precisione.** E' il metodo standard di trattare i numeri in Basic M24, se non altrimenti specificato. Il nome della variabile che li contiene puo' avere il simbolo [!], ma se omesso e' sottinteso.
Il campo di definizione, con una significativita' massima di 7 cifre, e' compreso tra 10^(-38) e 10^(+38). Si noti pero' che un numero di piu' di sette cifre viene memorizzato e visualizzato con un massimo di sette cifre (con arrotondamento). Come abbiamo gia' visto nel primo capitolo un numero come 1.7818577 viene arrotondato a 1.781858; la stessa cosa avviene per il numero dieci milioni di volte superiore 17818577 che pero' viene rappresentato in notazione esponenziale con 1.781858E+07, dove E sta per Esponenziale, ed E+07 sta per settima potenza positiva di 10, che vale appunto 10 milioni, un numero con sette posizioni decimali.
Dopo la trasformazione, se volessimo riavere il numero impostato in notazione decimale , otterremmo soltanto 17818580. Con la stessa notazione, che consente di conservare le cifre significative, si possono esprimere numeri molto piccoli; ad esempio 1.234567E-9 equivale a 0.000000001234567.

REALI **in doppia precisione.** E' il modo di registrare i numeri che occupa maggior memoria e che impiega piu' tempo nel fare i calcoli. Per il resto si possono fare considerazioni simili a quelle appena espresse per i numeri in semplice precisione. La differenza, naturalmente, sta nel fatto che possiamo trattare numeri piu' grandi (fino a 16 cifre) senza farli passare in notazione esponenziale.
Il campo di variabilita' va da 10^(-308) a 10^(+308). La variabile con la qualifica di contenitore di numeri in doppia precisione **deve** avere in coda al nome il simbolo del cancelletto [#].
Si noti che anche le costanti possono avere in coda la qualifica di doppia precisione; anzi, questo e' l'unico modo per introdurre un numero di 16 cifre e non vederselo decimare a uno di sette cifre con arrotondamento. Ad esempio introducendo a#=12345678901234567 con la manovra ?a [CR] si ottiene 1.234567890123457D+16.

6.3 LE VARIABILI CON INDICE

Contrapposte alle variabili fin qui considerate, si distinguono con il nome di **variabili con indice** quelle che raggruppate insieme, ma tutte dello stesso tipo (intere, semplice precisione, doppia precisione...), sono individuate dalla posizione che occupano in una certa matrice. Tale posizione, o indice, e' quindi un numero che identifica una certa variabile nel gruppo e viene posta tra parentesi, come argomento, suffisso, del nome della variabile. Il primo esempio riportato in figura 6.2 e' una variabile a un solo indice, di nome V( ), detta anche **vettore** o matrice unidimensionale.

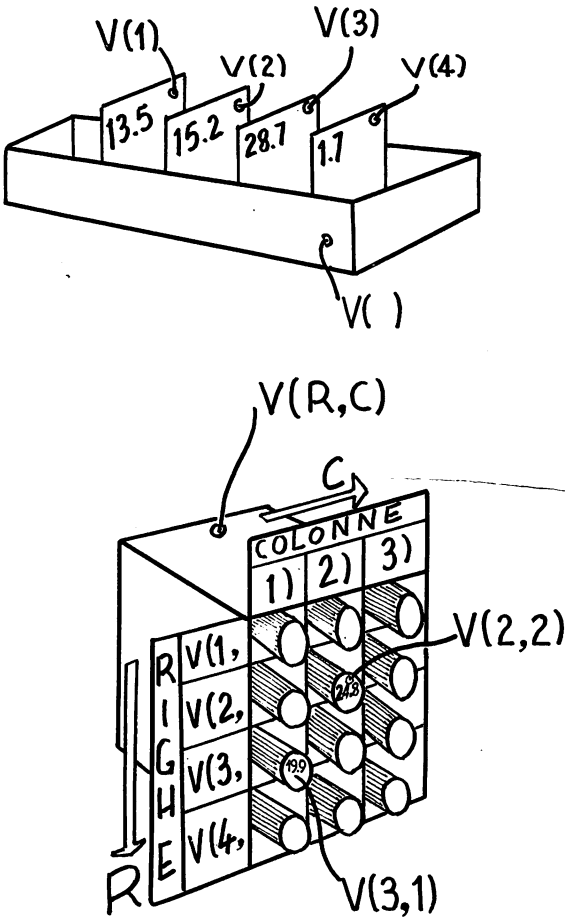



Figura 6.2

Se vogliamo che la matrice V( ) contenga 4 elementi ciascuno dei quali possa memorizzare un numero allo stesso modo di una variabile normale, basta scrivere quattro opportune relazioni di assegnazione, come nell'esempio seguente:

```
V(1)=13.5
V(2)=15.2
V(3)=V(1)+V(2)
V(4)=V(2)-V(1)
```

L'altro tipo di variabile con indice, illustrato nella figura 6.2, e' un esempio di matrice bidimensionale, detta anche matrice rettangolare, composta di 4 righe e di 3 colonne. Con istruzioni simili a quelle viste per la lista dei quattro elementi del vettore precedentemente considerato possiamo, ad esempio, assegnare valori alle variabili V(2,2) e V(3,1) con le relazioni V(2,2)=24.8 e V(3,1)=19.9.
In questo caso la variabile V( ) contiene 12 elementi, ma potrebbe avere una dimensione qualsiasi, purche' vi sia sufficiente spazio in memoria. Ad esempio, con riferimento alla figura 6.2, se ogni tubo contenesse, a sua volta una propria lista di 5 elementi, la matrice sarebbe tridimensionale e conterrebbe (4x3x5)= 60 elementi. Per individuare il primo, di posto (1,1,1), basta chiamare la variabile V(1,1,1); quello designato V(2,3,5) e' l'ultimo della lista contenuta nel tubo individuato dalla riga 2, colonna 3. L'argomento viene sviluppato nel paragrafo 9.5, in tema di ordinamenti.

6.4 LE FUNZIONI GIA' DEFINITE IN M24

A questo tipo appartengono le funzioni prefabbricate in M24.

6.4.1 FUNZIONI NUMERICHE ESPONENZIALI

EXP(x) Tale funzione consente di calcolare la potenza x in base e (2.71828). La variabile x deve essere non superiore a 88.0296 se in semplice precisione, oppure 88.02968979 quando si opera in doppia precisione.

SQR(x) Fornisce la radice quadrata di x, per x non inferiore a zero

LOG(x) Da' il logaritmo in base **e** di x: cioe' fornisce l'esponente da assegnare alla base **e** per ottenere x. La variabile x deve essere maggiore di zero

Segue il programma **espo** per il calcolo delle tre funzioni, con introduzione del valore della variabile x.

```
10 'espo: funzioni esponenziali
15 CLEAR:CLS:SCREEN 1:KEY OFF
20 INPUT "X= ",X
30 PRINT "e^x= "EXP(X)
40 PRINT "x^(-1/2)= "SQR(X)
50 PRINT "log(x)= "LOG(X)
55 INPUT"",A
60 GOTO 15
```

```
X= 1
e^x=  2.718282
x^(-1/2)=  1
log(x)=  0
```

6.4.2 FUNZIONI TRIGONOMETRICHE

SIN(x), **COS**(x), **TAN**(x). Forniscono rispettivamente il seno , il coseno e la tangente di un angolo x, dove x e' espresso in radianti. Si ricorda che, se la misura dell'angolo e' in gradi sessagesimali, bisogna prima trasformarlo in radianti dividendo i gradi per il fattore 57.2558. Tale fattore e' il rapporto 180/PG, dove **PG**(pgreco)=**3.1415926535**...(Il radiante e' l'angolo al centro di un arco la cui lunghezza rettificata e' pari al raggio). Infatti dalla Geometria elementare, indicando con **alfa** la misura in gradi di un arco e con **l** e **r** la lunghezza di questo arco e la lunghezza del suo raggio rispetto alla stessa unita', si ha:

l=alfa r (PG)/180 e alfa=l 180/(PG)r

Se, in particolare, si assume per unita' di lunghezza il raggio, si ha la lunghezza dell'arco riferita al raggio, la quale come l'ampiezza, e' indipendente dal raggio; e allora, indicando con **a** questa lunghezza, si ha evidentemente

a= alfa (PG/180) e alfa=a (180/PG)

Cosi', ad esempio,

si ha a=0.000004848136811... se alfa= 1"

(dove 1" e' il minuto secondo, trecentosessantesima parte del grado, che, a sua volta e' la trecentosessantesima parte dell'angolo giro).

In particolare, se a=3(PG)/2 , (PG) , (PG)/2 , (PG)/4 , (PG)/16 ,.. .. si ha rispettivamente alfa=270 gradi, 180 gradi, 90 gradi, 45 gradi, 11 gradi e 15',...

Da quanto detto si intuisce come le funzioni trigonometriche di un arco non dipendano dalla lunghezza, ma solo dall'ampiezza, o dalla misura riferita al raggio di questo arco e, di conseguenza, come due archi simili, cioe' con angoli al centro uguali, abbiano le stesse funzioni trigonometriche.

ATN(x) fornisce l'arcotangente di x. Il risultato, in radianti, e' compreso tra -PG/2 e +PG/2. Per avere la misura in gradi occorre moltiplicare il risultato per 180/PG= 57.2958...

6.4.3 FUNZIONI GENERATRICI DI NUMERI CASUALI (RANDOM)

La parola chiave in Basic per generare dei numeri random e' **RND**. La relazione

Y=RND

assegna a y un valore casuale compreso tra 0 e 1. In realta', si tratta di numeri casuali estratti da un deposito in memoria che ne contiene circa un milione. La sequenza quindi si ripete quando tutti

i numeri sono stati utilizzati.

Per cambiare la sequenza dei numeri ogni volta che tale istruzione viene eseguita occorre premettere l'istruzione RANDOMIZE **i** il cui argomento **i** costituisce il punto di ingresso in quel deposito, o, come si dice, il seme della generazione (random seed). Se l'istruzione viene scritta senza tale numero, sara' la macchina a chiedervelo. Altrimenti puo' essere accoppiato a **qualche** numero interno di macchina. L'artificio, che piu' comunemente viene utilizzato, e' quello di impiegare il contenuto dell'orologio di macchina, per esempio i **minuti secondi.**

Per far cio' dovremmo utilizzare alcune funzioni di trattamento delle stringhe di cui al momento non abbiamo ancora parlato.
Con M24, tuttavia, esiste una variante dell'istruzione (RANDOMIZE TIMER), che fa appunto cio' automaticamente.

Per far un po' di esperienza, comunque, cominceremo dal seguente programma di generazione, di nome ran62, che chiede all'esterno il seme della generazione, attraverso la suddetta RANDOMIZE (senza argomento).

```
10 'ran62 :numeri casuali con randomize
20 CLEAR '      aggiornata dall'esterno
25 CLS:KEY OFF:SCREEN 3
30 INPUT;"n.elem.",N
40 PRINT
50 RANDOMIZE
60 FOR K=1 TO N
70 X=RND
80 PRINT X,100*X,INT(100*X)
90 NEXT
100 INPUT"",W
110 GOTO 20
```

Il programma, dopo aver ottenuto il numero (N) degli elementi da estrarre (INPUT alla linea 30) e il seme della generazione (un numero intero) a seguito della Randomize, cicla N volte X=RND e stampa gli N numeri casuali, gli stessi moltiplicati per un fattore 100 e il risultato dell'arrotondamento operato dalla funzione INT(X) per troncamento della parte decimale.

Alcune utili espressioni per ottenere la generazione di numeri casuali che cadano in un certo intervallo si deducono dalla formula generale:

X= INT (EA-EB+1)*RND+EB

dove EA e EB sono, rispettivamente, gli estremi alto e basso dell'intervallo dei numeri interi da cui vogliamo estrarre numeri casuali.

6.4.4 LE FUNZIONI DI STRINGA

Ora che sapete usare le istruzioni Basic per i numeri, vediamo di stabilire un parallelo con le stringhe, di cui e' opportuno ripetere le definizioni gia' date nei precedenti paragrafi.

Una stringa e' una successione di codici, come quelli che compaiono nella tabella ASCII, vedi oltre in figura 6.3, dalla quale abbiamo estratto per comodita' di consultazione alcuni spezzoni semplificati riportandoli qui di seguito.

DEC.	ESADEC.	CODICE
7	07	BEL
8	08	BS
9	09	HT
10	0A	LF
11	0B	VT
12	0C	FF
13	0D	CR

DEC.	ESADEC.	CODICE
65	41	A
66	42	B
67	43	C
97	61	a
98	62	b
99	63	c

Figura 6.3

Si puo' notare che alcuni dei codici ASCII corrispondono a caratteri stampabili, mentre altri, non stampabili, corrispondono a funzioni di macchina; tra queste, ad esempio, l'avvisatore acustico che ha come codice la sigla BEL. Corrispondenti a questa sigla la tabella ha due entrate: quella decimale (7) e quella esadecimale (07), Ricordiamo che la notazione esadecimale si riferisce alla numerazione in base 16 e pertanto il riporto avviene a 16: cioe' dopo il 9 non viene il 10, ma 0A e dopo il 0F viene il 10, e cosi' via.

Analogamente la lettera A (maiuscola) ha come entrata decimale 65 e come entrata esadecimale 41. A questo punto una domanda e' d'obbligo: E' possibile confezionare stringhe fatte anche di caratteri non stampabili? Si'! Il motivo lo scopriremo quando impareremo a conoscere meglio le funzioni di stringa. Intanto ricordiamo che per stampare una stringa di caratteri stampabili e' sufficiente racchiuderla tra due virgolette e affidarla alle cure di una PRINT.

Cosi' una PRINT "Aa" fa stampare Aa. Ma si puo' ottenere lo stesso risultato utilizzando una funzione di stringa particolare che ha la seguente struttura:

```
CHR$ (D)
```

dove D e' l'entrata decimale nella tabella ASCII corrispondente ai codici desiderati. Cosi'

```
PRINT CHR$(65) CHR$(97)          fa stampare ancora        Aa
```

Questo secondo modo e' senza dubbio meno pratico del precedente per visualizzare codici stampabili, ma e' l'unico che consente di estrarre, in forma parametrica i caratteri della tabella ASCII e in particolare quelli corrispondenti alle funzioni di sistema.

Tra queste alcune sono di uso comune e conviene ricordarle a memoria

Print CHR$(7)	fa emettere un bip all'avvisatore acustico;
Print CHR$(10)	fa scrollare (scorrere verticalmente) una linea del video;
Print CHR$(13)	fa spostare il cursore a capo (la posizione piu' a sinistra della riga);
Print CHR$(10) CHR$(13)	equivale a [CR].

Per inciso, e' opportuno notare che, in modo immediato, si possono generare da tastiera tutti i caratteri della tabella facendo uso del **tasto speciale ALT**: e' sufficiente, infatti, tenerlo premuto mentre si introduce il numero corrispondente all'entrata decimale.

La funzione duale di CHR$ e' **ASC**(stringa). Con tale funzione si ottiene il valore dell'entrata decimale nella tabella ASCII del primo carattere della stringa argomento.

Segue ora il programma **forasci** con il quale e' possibile stampare i simboli corrispondenti alle entrate decimali della tabella ASCII.

Si noti che alla linea 22 abbiamo filtrato l'entrata 12 che corrisponde alla pulizia dello schermo: in caso contrario avremmo perduto la stampa dei primi simboli 11. Nella figura 6.3 la stampa dei simboli ASCII organizzati in tabella 13 x 20.

```
10 'forasci: stampa ASCII
15 CLS:KEY OFF:SCREEN 3
20 FOR D=0 TO 255
22 IF D=12 THEN 50
40 A$=CHR$(D):PRINT " "A$;
50 NEXT
```

1																			20
☺	☻	♥	♦	♣	♠		◘						♫	☼	►	◄	↕	‼	¶
§	▬	↨	↑	↓	→	←						!	"	#	$	%	&	'	(
)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<
=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c	d
e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
y	z	{	¦	}	~	⌂	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î
ì	Ä	Å	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ	á
í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»	░	▒	▓	│	┤
╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐	└	┴	┬	├	─	┼	╞	╟	╚
╔	╩	╦	╠	═	╬	╧	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄
▌	▐	▀	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩	≡
±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■						

254

Figura 6.3

I successivi programmi **asci** e **ascitab** forniscono i codici corrispondenti alle entrate decimali della tabella e viceversa.

```
10 'asci:stampa i simboli ASCII a partire
20 '     dai corrispondenti car. decimali.
25 CLEAR:CLS:KEY OFF:SCREEN 3
30 INPUT; "    D= ",D
40 A$=CHR$(D)
50 PRINT "  ---> "A$
60 GOTO 30
70 END
```

```
10 'ascitab: corrispondenza ASCII
20 KEY OFF:CLS:PRINT"per ottenere ASCII(a)...DECIM.(d)-->"
25 R$=INPUT$(1):PRINT"                                Risposta---->"R$
30 IF R$="a" GOTO 70 ELSE IF R$="d" GOTO 40 ELSE GOTO 20
40 PRINT " <car. dec.> corrispondente al";
50 INPUT;"  simbolo ASCII=",ASCI$
55 IF ASCI$="]]" GOTO 120
60 PRINT "<"ASC(ASCI$)">":GOTO 40
70 PRINT "  simbolo ASCII corrispondente al";
80 INPUT;"  num.decim.=",CC:IF CC>255 GOTO 120
90 IF CC > 255 GOTO 120
100 PRINT "<"CHR$(CC)">":GOTO 70
120 PRINT
130 GOTO 20
```

Infine il programma **tast** fornisce, per ogni tasto battuto singolarmente o in accoppiata con i tasti speciali CTRL o SHIFT, il valore della corrispondente entrata ASCII.

```
10 'tastolab:laboratorio per tasti
20 CLS:KEY OFF:SCREEN 2
30 PRINT "PREMENDO UN TASTO SI OTTIENE IL NUMERO D'ORDINE
40 PRINT "DELLA CORRISPONDENTE ENTRATA DECIMALE IN TAB.ASCII
50 QQ$=INPUT $(1)
60 KK=ASC(QQ$)
70 PRINT "-"KK;
80 GOTO 50
```

Sul concetto di **stringa** si tengano presenti le seguenti proprieta':

- la **costante** stringa e' una successione di caratteri ASCII di lunghezza variabile (lunghezza massima 255 caratteri) racchiusa tra **virgolette** ["];

- la **variabile** stringa e' un contenitore di stringhe il cui nome **deve** terminare con il simbolo $ (dollaro);

- per assegnare un valore a una variabile stringa, come per i numeri, usiamo le relazioni di assegnazione, tipo A$="ALFA" con la quale carichiamo la costante stringa "ALFA" nella variabile stringa A$.
  (Attenzione: A$=ALFA da' errore perche' non c'e' compatibilita' di tipo di variabile tra primo e secondo membro: infatti ALFA senza virgolette e' a tutti gli effetti una variabile reale in semplice precisione).
  Notiamo inoltre che per introdurre un numero, ad esempio 2, nella variabile A$, occorre scrivere A$="2";
- per azzerare la variabile stringa occorre caricarvi una stringa nulla (""), che corrisponde allo zero dei numeri;
- tra gli operatori numerici, l'unico che e' possibile usare e' il segno [+], ma naturalmente tra stringhe il concetto di somma e' diverso da quello tra numeri, e equivale a una concatenazione.

Notiamo, infine, che si possono usare operatori di relazione come = < > e le loro combinazioni, dove il significato e' ora quello di coincidenza, precede e segue. Il criterio di confronto tra due stringhe e' il seguente.

Si confrontano i caratteri a coppie, partendo da sinistra. Se, a un certo punto, dopo una serie di coppie di caratteri uguali, si incontrano due caratteri diversi, si valutano i relativi codici ASCII di tali caratteri, interpretandoli come numeri binari senza segno: la stringa minore e' quella cui appartiene il codice minore.

In particolare, se, dopo una serie di coppie di caratteri uguali, una stringa finisce e l'altra no (si noti che gli spazi o blank sono significativi in una stringa), quella minore e' < rispetto a quella piu' lunga. Ad esempio, consideriamo le due stringhe:

"aab" e "aa "

Esse differiscono per il terzo carattere, che per la prima e' una b e per la seconda e' uno spazio .I corrispondenti codici ASCCI sono:

b corrisponde all'entrata decimale 98
  corrisponde all'entrata decimale 32

Poiche' 32<98, "aa " < "aab"

Il programma **strincon** consente di sperimentare confronti tra stringhe di caratteri.

```
10 'strincon:confronto tra stringhe
20 CLS:KEY OFF:SCREEN 2
30 INPUT "INTRODURRE UNA PRIMA STRINGA  --> ",Q1$
45 INPUT "INTRODURRE LA SECONDA STRINGA -->",Q2$
60 IF Q1$<Q2$ THEN PRINT Q1$"<"Q2$:PRINT:GOTO 30
70 IF Q1$>Q2$ THEN PRINT Q1$">"Q2$:PRINT:GOTO 30
80 PRINT Q1$"="Q2$
90 PRINT:GOTO 30
RUN
```

```
INTRODURRE UNA PRIMA STRINGA  --> a a
INTRODURRE LA SECONDA STRINGA -->aa
a a<aa

INTRODURRE UNA PRIMA STRINGA  --> a 1
INTRODURRE LA SECONDA STRINGA -->a1
a 1<a1
```

Per introdurre valori stringa da tastiera sono disponibili i seguenti tre tipi di istruzioni:

1) Il primo tipo di istruzione e' un'emanazione di quella gia' conosciuta per la variabili numeriche. Ad esempio, per introdurre ALFA da tastiera nella variabile A$ basta scrivere l'istruzione:

   INPUT A$

Come per le variabili numeriche e' possibile la variante

   INPUT"A=",A$

che evitera' il segnale [?] visualizzando A=

2) Quando si vuole introdurre una stringa di lunghezza controllata si usa invece la seguente istruzione:

   A$=INPUT$(n)

dove n e' il numero di caratteri di cui e' costituita la stringa che si vuole introdurre nella variabile assegnata e che non vengono visualizzati sul video.

E' interessante notare che per questo tipo di INPUT non e' necessario battere [CR]: se, ad esempio, n=4 e si vuole introdurre BETA, alla digitazione del quarto carattere (A) l'introduzione della variabile stringa A$ ha termine. Per visualizzare quanto introdotto occorre far seguire una PRINT A$.

3) Per impieghi particolari e quando interessi caricare nella variabile assegnata soltanto il primo dei caratteri introdotti da tastiera, che anche in questo caso non vengono visualizzati, esiste il seguente terzo tipo di istruzione:

   A$=INKEY$

La differenza piu' rilevante tra la INKEY$ e la INPUT$ e' che mentre quest'ultima e' di tipo sospensivo, cioe' quando il programma la incontra non va avanti finche' non viene soddisfatta, la INKEY$, invece, viene comunque eseguita e se non abbiamo impostato nulla da tastiera la macchina assegna stringa nulla "" alla variabile A$.

Seguono ora le descrizioni illustrate delle piu' importanti funzioni di stringa.

LEFT$(A$,n) dove A$ e' una variabile stringa dalla quale vogliamo estrarre i primi **n** caratteri piu' a sinistra (vedi figura 6.4).

RIGHT$(A$,n) Questa istruzione e' simile alla precedente, ma estrae i primi **n** caratteri a partire dall'estremo destro della stringa A$ (figura 6.4).

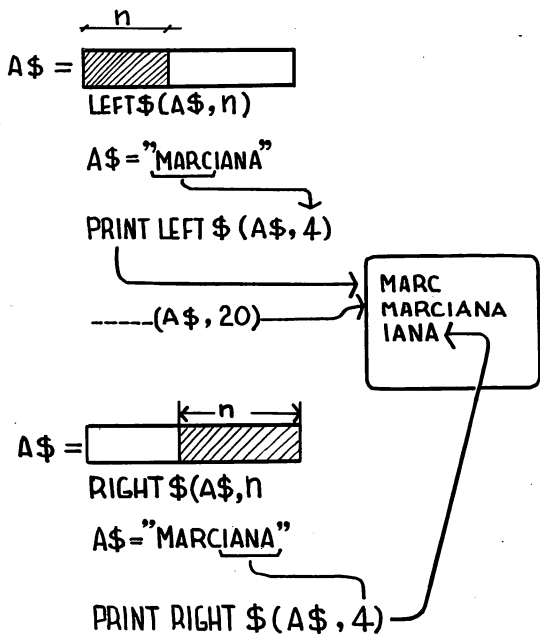



Figura 6.4

LEN (A$)

Calcola la lunghezza della stringa A$ (vedi figura 6.5)

Figura 6.5

STRING$(v,d) Fabbrica una stringa tutta costituita di caratteri uguali al codice ASCII corrispondente all'entrata decimale **d** della tabella, ripetuti **v** volte (figura 6.6).

Figura 6.6

MID$(S$,F,L) Questa istruzione ha due versioni. Nella prima **estrae** da una stringa denominata S$ una sottostringa di lunghezza L a partire dalla posizione F (vedi figura 6.7)

Figura 6.7

Ovviamente tutte le funzioni fin qui descritte devono essere incorporate in una PRINT se vogliamo visualizzarle; se, invece, desideriamo semplicemente assegnare dei valori a una variabile bastera' scrivere le corrispondenti relazioni:

B$=LEFT(A$,n), C$=RIGHT(A$,n), Z$=STRING$(v,d),...Y=MID$(S$,F,L)..
.etc.
L'altra versione della MID e' costituita da una relazione di assegnazione dove il primo membro ha la stessa struttura utile a individuare una certa sottostringa a partire da una stringa S$ di partenza; nel secondo si dichiara, tra virgolette, una certa costante stringa che andra' a prendere il posto della sottostringa.

Con questa seconda versione la MID consente dunque una sostituzione -in tutto o in parte- di una data stringa con un'altra (vedi ancora la figura 6.7).
Con il programma **camid** si fa un'indagine completa sulle possibilita' della **mid** e sulle istruzioni **left right e asc.**

```
10 'camid:capire la mid
20 CLS:CLEAR
30 INPUT; "POSTO=",P
40 INPUT ";  LUNG.=",L
50 INPUT "S=",QQ$
60 SS$=MID$(QQ$,P,L)
70 K%=ASC(SS$)
80 PRINT "sottos.=";SS$
90 PRINT"primo estratto MID=";K%
100 M$=LEFT$(SS$,1):PRINT "primo da sinistra=";M$;"  ";
110 KS%=ASC(M$)
120 PRINT "E.ASCII=";KS%
130 D$=RIGHT$(SS$,1):PRINT "primo da destra=";D$;"    ";
140 KD%=ASC(D$)
150 PRINT "E.ASCII=";KD%
160 PRINT:GOTO 30

RUN
POSTO=6;  LUNG.=3
S=DOPPIE ENTRATE
sottos.=E E
primo estratto MID= 69
primo da sinistra=E  E.ASCII= 69
primo da destra=E    E.ASCII= 69
```

Segue, infine, un piccolo esempio di impiego della MID e della LEN nel programma **mid.**

```
10 'mid: trattamento parole
15 CLS:SCREEN 1:KEY OFF
20 CLEAR
30 INPUT"la parola ---- --> ",S$
40 C$=""
50 L=LEN(S$)
60 FOR C=L TO 1 STEP-1
70 C$=C$+MID$(S$,C,1)
80 NEXT C
90 PRINT"al rovescio e' --> "C$
100 PRINT:GOTO 30
```

```
la parola ---- --> PAUL SAMUELSON
al rovescio e' --> NOSLEUMAS LUAP

la parola ---- --> DANTE ALIGHIERI
al rovescio e' --> IREIHGILA ETNAD
```

Per finire, esaminiamo la piu' sofisticata delle funzioni di stringa:

INSTR(F,S$,C$) Questa funzione fornisce la posizione (il **numero** di posizione) in cui si trova l'inizio della sottostringa C$ in una data stringa S$. Il parametro F nella funzione serve come base di riferimento da cui iniziare (o continuare) la ricerca di C$. La posizione di cio' che si e' trovato e' pero' sempre numerata dall'inizio (vedi esempio parole3); se F viene omesso implicitamente la macchina assume F=1, cioe' la base e' la posizione del primo carattere della stringa S$ (vedi figura 6.8).

In tale figura riportiamo anche il listato del programma **parole3**, che consente di analizzare un testo attraverso la funzione INSTR. Sia data la stringa S$: vogliamo sapere quante volte una certa parola, un connettivo preposizionale, una sottostringa (prefisso, suffisso, etc.) compaiono e in quali posizioni.

La linea 110 conta gli esiti positivi della ricerca. La 120 riposiziona la freccia F sulla posizione dell'ultimo reperto: appena non ci sono piu' reperti V=0 e la 90, rivelando tale condizione, fa saltare alla linea 140 per la stampa dei risultati. La scelta alla linea 170 consente di ripetere subito sullo stesso testo ricerche diverse. Questo e' solo uno dei piu' semplici esempi di analisi di un testo: lascio alla vostra fantasia immaginare tutte le altre infinite e ben piu' raffinate ricerche che si possono fare con la combinazione delle funzioni di stringa descritte.

6.4.5 FUNZIONI OROLOGIO

Una delle grandezze fondamentali per la misurazione dei fenomeni fisici e' il tempo. Come potremmo trascurarla usando un elaboratore?

Non e' che vogliamo usare M24 come un orologio, ma poiche' questo fa parte dei suoi dispositivi interni impariamo ad utilizzarlo indicando anche i momenti in cui conviene farlo. Procediamo, quindi, con ordine, spiegando le principali caratteristiche dell'orologio.

- Funziona anche a macchina spenta, grazie a una batteria tampone allocata al suo interno.

- Il tempo e' espresso in ore, minuti, secondi, (hh:mm:ss).Il valore di partenza standard assegnato da MS-DOS, all'accensione di M24, e' quello corrente dopo l'ultimo riposizionamento, ma se vogliamo puo' essere rimesso ad un valore qualsiasi con il comando **time** di MS-DOS.

```
LIST
10 'parole3: ricerca di una chiave su stringa
20 CLEAR
25 LINE INPUT"stringa in esame= ",S$
30 LINE INPUT"stringa di ricerca= ",C$
50 P=0
60 F=1
70 PRINT"posizioni dei reperti: ";
80 V=INSTR(F,S$,C$)
90 IF V=0 GOTO 140
100 PRINT V;
110 P=P+1
120 F=V+1
130 GOTO 80
140 PRINT:PRINT
150 PRINT "numero di reperti= "P
160 PRINT"---------------------------------------------
170 PRINT:PRINT"premi <s> se vuoi continuare"
175 A$=INPUT$(1):IF A$="s" THEN GOTO 30 ELSE GOTO 20
Ok
RUN
stringa in esame= Nel mezzo del cammin di nostra vita
stringa di ricerca= m
posizioni dei reperti:  5  17  18

numero di reperti=  3
---------------------------------------------

premi <s> se vuoi continuare
stringa di ricerca= mm
posizioni dei reperti:  17

numero di reperti=  1
---------------------------------------------

premi <s> se vuoi continuare
stringa di ricerca=
posizioni dei reperti:  4  10  14  21  24  31

numero di reperti=  6
---------------------------------------------

premi <s> se vuoi continuare
stringa di ricerca= st
posizioni dei reperti:  27

numero di reperti=  1
---------------------------------------------

premi <s> se vuoi continuare
```

Figura 6.8

Se stiamo lavorando in ambiente MS-DOS (segnale a> presente) e vogliamo rimettere l'orologio alle cinque e un quarto del pomeriggio, basta impostare **time 17:15:00 (CR)**. Per inciso, con analoga procedura possiamo impostare la data: se scriviamo **date 12/25/85 (CR)**, M24 prende nota anche della data del 25 dicembre 1985. Da quest'istante i dispositivi interni di M20 per la misura del tempo evolvono a partire dai valori immessi con i comandi date e time.
Ripetiamo che, in assenza di manovre, la data e il tempo hanno i valori correnti che provengono dall'ultimo aggiornamento.

- Per sapere che ora e' basta impostare in Basic **PRINT TIME$ (CR)**. Oppure **?time$** [CR]. Analogamente, per la data si scrive: PRINT DATE$ [CR].

- Il tempo trascorso tra due particolari momenti di macchina si puo' ottenere in due modi diversi. Il primo, piu' semplice, consiste nell'azzerare l'orologio in corrispondenza del primo evento e poi nel rileggerlo al verificarsi del secondo evento: in tal modo pero' nel rimettere a zero l'orologio abbiamo perso la nozione dell'ora attuale di macchina.
Il secondo modo, senza rimessa a zero, consiste nel fare le due letture in corrispondenza dei due eventi e poi farne la differenza.

Proviamo a scrivere un programma di due sole istruzioni, come il seguente

```
100 PRINT TIME$
110 GOTO 100
```

Il risultato sara' una sequenza senza fine di valori estratti dall'orologio che scorrono verticalmente nelle prime colonne del video. Se mettete al termine della 100 un punto e virgola, vi invaderanno tutti i 2000 caratteri del video.

Immaginiamo ora un artificio per ridurre il numero delle letture e inseriamo dei tempi morti tra una lettura e la successiva.

Utilizziamo a tal fine quel mattoncino di Basic (programma forat) che abbiamo fabbricato al paragrafo 5.1 e che ora ritroviamo condensato alla linea 60 nel seguente programma denominato **orol**:

```
10 'orol: lettura orologio
20 SCREEN 1:CLEAR:CLS
30 INPUT "t= ",T
40 't=2600 per 1 sec di scansione
50 LOCATE 12,15:PRINT TIME$
60 FOR K=1 TO T:NEXT
65 QQ$=INKEY$:IF QQ$="/" THEN 20
70 GOTO 40
```

Tale programma contiene anche una INPUT per tarare la FOR...NEXT e una LOCATE 12,15 che ad ogni riciclo della GOTO riporta l'immagine al centro dello schermo a bassa risoluzione, cioe' alla riga 12, colonna 15.

Per tarare a un secondo il periodo tra una lettura e la successiva

il valore di t deve essere circa 2600; per valori inferiori l'ora verra'aggiornata al ritmo di una volta al secondo; per valori superiori vedremo che l'orologio si aggiorna ogni volta che il contatore t si azzera.
Il seguente programma **rimol**, invece, all'istruzione 35 contiene un'assegnazione alla funzione di macchina TIME$ che e' in collegamento con l'orologio (altro modo per rimetterlo a zero, invece di usare il comando **time** di MS-DOS).

```
10 'rimol: lettura orologio
20 CLEAR ' con rimessa iniziale
30 CLS:INPUT "t= ",T
35 TIME$="00:00:00
40 CLS:LOCATE 12,15
50 PRINT TIME$
60 FOR K=1 TO T:NEXT
70 GOTO 40
```

6.5 ESPRESSIONI ALGEBRICHE

Facciamo pratica di calcolo di espressioni in linguaggio Basic con qualche semplice esempio di trigonometria. Con riferimento alle grandezze illustrate nella figura 6.9, esempi classici di impiego di funzioni trigonometriche sono quelle che compaiono nelle seguenti relazioni:

```
AREA = 1/2 (S T) sen(RA/C)
  SA = arctg ( S sen(RA)/C / (T - S cos(RA/C))) C
  TA = arctg ( T SEN(RA/C) / (S - T cos(RA/C))) C
   R = (T^2 + S^2 - 2S T cos(RA/C))^( 1/2)
```

dove R, S e T sono le misure dei lati di un triangolo e RA, SA e TA i rispettivi angoli opposti espressi in gradi e C il coefficiente per passare da gradi a radianti.

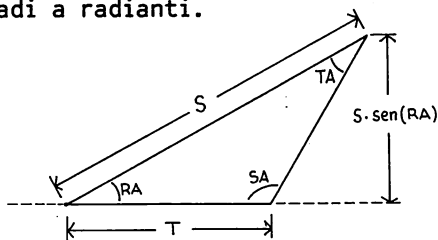



Figura 6.9

Con le relazioni sopra indicate si possono calcolare tutti gli elementi di un triangolo,conoscendo solo le misure S e T di due lati e l'angolo [RA] tra loro compreso.
Segue il programma **trigo** per la risoluzione del problema.

```
10 'trigo: calcolo del triangolo, dati due lati
15 KEY OFF
20 CLEAR:SCREEN 2:CLS'  e l'angolo compreso
30 INPUT"lato S= ",S
40 INPUT"lato T= ",T
50 INPUT;"angolo RA (in gradi)= ",RA
60 PG=3.1415926535# 'pgreco
```

```
70 C=180/PG        'coefficiente divisore
80 '      per passare da gradi a radianti
90 AREA=.5*S*T*SIN(RA/C)
100 SA=ATN(S*SIN(RA/C)/(T-S*COS(RA/C)))*C
110 TA=ATN(T*SIN(RA/C)/(S-T*COS(RA/C)))*C
120 R=SQR(T^2+S^2-2*S*T*COS(RA/C))
130 PRINT:PRINT
150 PRINT "lato R= "R
155 PRINT"..area= "AREA
156 PRINT
160 PRINT "    SA= "SA
170 PRINT "    TA= "TA
180 INPUT"",W
190 GOTO 20
200 LIST
RUN
lato S= 5
lato T= 5
angolo RA (in gradi)= 90

lato R=  7.071069
..area=  12.5

    SA=  45
    TA=  45

lato S= 5
lato T= 5
angolo RA (in gradi)= 60

lato R=  5
..area=  10.82532

    SA=  60
    TA=  60
```

6.6 FUNZIONI DEFINIBILI DALL'UTENTE

Quando si deve utilizzare piu' volte un'espressione complessa puo' essere utile fabbricare una funzione simbolica corrispondente che ha in evidenza come argomento solo la variabile da cui dipende. Questa nuova funzione sintetica puo' essere usata nel programma come una normale funzione di sistema, anche se la sua definizione vale solo nell'ambito di un programma. L'istruzione Basic che consente di far questo e' la **DEF FN.** Ad esempio, se scriviamo:

```
DEF FN DELTA(x) = (2 + X)^3 - 5*X^3
```

Nel momento in cui vogliamo impiegare quell'espressione al secondo membro, sara' sufficiente richiamare in sua vece la funzione con il nome definito al primo membro. Cosi', se vogliamo assegnare alla variabile W il valore che assume DELTA per X = 2,scriveremo:

```
W=FN DELTA(2)
```

La macchina sostituisce X=2 nell'espressione analitica originaria, ottenendo:

(2+2)^3-5*2^3=4^3-5*2 3=64-40= 24,

e assegna il valore 24 alla variabile W.
Segue ora un semplice programma di matematica finanziaria nel quale e' inclusa una funzione definita dall'utente. Supponiamo di voler calcolare, in **regime di capitalizzazione composta**:

1) il montante, cioe' il valore futuro del capitale C impiegato per N periodi, al tasso di interesse T; oppure,

2) il tempo necessario (espresso in N periodi) affinche' con un certo capitale iniziale C e gli interessi accumulati al tasso periodico T si ottenga un montante pari a M; oppure, ancora,

3) il tasso d'interesse T, per ottenere il montante M, impiegando un capitale C e per N periodi.

Questo genere di problemi si risolvono elaborando la formula generale del montante in regime di capitalizzazione composta:

1) M = C (1 + T)^N (^N sta per elevato a esponente N)

Il problema si risolve ricavando N dalla 1) con i seguenti sviluppi:

M/C = (1 + T)^N; log M - log C = N log (1 + T)

da cui

2) N = (log M - log C)/(log (1 + T)

Dalla 1) si ricava infine: (M/C)^(1/N) = (1 + T)

e quindi

3) T = ((M/C)^(1/N))-1

Se volete fare un programma che risolva tutti e tre i problemi, conviene avere le tre formule in tre diverse linee alle quali rinviare attraverso scelte iniziali: e' quanto accade nel programma **incomp** che riportiamo con il listato di alcune passate.

```
10 'incomp:i tre casi della capitalizzazione composta
15 CLEAR:SCREEN 2:CLS
16 PRINT"premere 1 per MONTANTE"
17 PRINT"  ''    2 per PERIODI"
18 PRINT"  ''    3 per TASSO"
20 INPUT"                          ----->",A
30 IF A=1 THEN GOSUB 100 ELSE IF A=2 THEN GOSUB 200
40 IF A=3 THEN GOSUB 300
50 PRINT:GOTO 16
100 INPUT "T  tasso(%) =" ,T
```

```
110 INPUT"N   (periodi)=" ,N
120 INPUT "C (cap. imp.)=" ,C
130 M= C*(1+T/100) ¯N
140 PRINT "montante=" ,M
150 RETURN
200 DEF FNLOG10(X)=LOG(X)/LOG(10)
210 INPUT"C=",C
220 INPUT"M=",M
230 INPUT"T=",T
240 N=(FNLOG10(M)-FNLOG10(C))/(FNLOG10(1+T/100))
250 PRINT"numero di periodi=",N
260 RETURN
300 INPUT "M=",M
310 INPUT "C=",C
320 INPUT"N=" ,N
330 T =100*(((M/C) ¯(1/N))-1)
340 PRINT "tasso=",T
350 RETURN
RUN
premere 1 per MONTANTE
  ''    2 per PERIODI
  ''    3 per TASSO
                              ----->1
T   tasso(%) =20
N   (periodi)=10
C (cap. imp.)=1
montante=      6.191737

premere 1 per MONTANTE
  ''    2 per PERIODI
  ''    3 per TASSO
                              ----->2
C=1
M=2
T=20
numero di periodi=            3.801784

premere 1 per MONTANTE
  ''    2 per PERIODI
  ''    3 per TASSO
                              ----->3
M=2
C=1
N=1
tasso=         100

premere 1 per MONTANTE
```

Come dicevamo il programma offre un esempio di utilizzo di funzioni interne definite liberamente dall'utente: alla linea 200, infatti, abbiamo definito la nuova funzione log(x)/log(10) col nome simbolico **LOG10(x)**. Questa funzione serve come moltiplicatore in tutti i termini logaritmici della 2) trasformandoli in base 10.

CAPITOLO
SETTE

ANCORA SULLA GRAFICA

In questo capitolo non ci interesseremo tanto di acquisire altre nozioni di informatica, ne' di trattare specifici impieghi gestionali di M24; lo scopo sara' piuttosto quello di usare l'elaboratore per stimolare quegli aspetti del pensiero che piu' si avvalgono del mezzo nuovo, come le attivita' creative.

Non e' escluso, infatti, che dall'impiego interattivo di un elaboratore, al di la' dei risultati immediati, si ottengano anche alcuni prodotti intermedi, frutto dell'interferenza tra lo strumento e la mente.

La forza nuova impressa all'analisi di un problema potrebbe derivare si' dall'eccezionale versatilita' di elaborazione, ma anche dalla definizione di nuovi rapporti tra tutti i nostri sensi per effetto dell'enorme dilatazione tecnologica. Uno dei contesti piu' chiari, a questo riguardo, e' il processo interattivo che consente di **verificare graficamente** un pensiero.

E cio' vale anche per le materie umanistiche, ad esempio nello studio della struttura di una lingua. Sembra quasi che torniamo, stranamente, alle origini dell'evoluzione del linguaggio, a quell'ideogramma in cui vista e significato erano intimamente congiunti!

7.1 GRAFICI DI FUNZIONI

Prima di esaminare alcuni programmi per **"graficare"** funzioni matematiche, riprendiamo ancora l'istruzione WINDOW gia' studiata nel paragrafo 5.4.

Questa istruzione e' fondamentale per la rappresentazione di grafici di funzioni, perche' consente di fissare il campo di manovra della variabile dipendente e fa si' che la curva venga tracciata nella finestra fisica in un determinato rapporto con la rappresentazione delle punteggiate x=0 (asse Y) e y=0 (asse X).

Facciamo il caso semplice di una retta che passa per i punti:P(0,1) e Q(1,0). Gli assi cartesiani di riferimento, in notazione Basic, siano i seguenti (vedi figura 7.1):

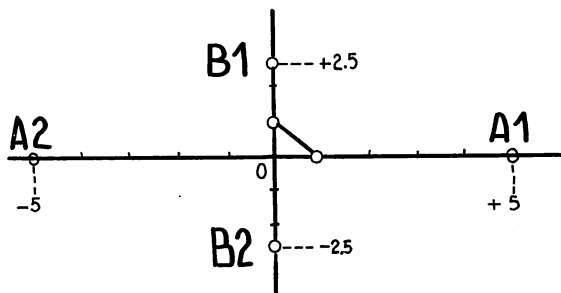



Figura 7.1

Il programma che traccia questa retta, denominato **graf**, e' il seguente:

```
10 'graf: prove grafiche
20 CLS:KEY OFF
25 INPUT "RISOL =",Z
28 SCREEN Z
60 LINE(-5,0)-(5,0)
100 LINE(0,-2.5)-(0,2.5)
110 LINE(0,4)-(4,0)
130 W$=INPUT$(1)
140 GOTO 20
```

Il risultato,date le ridotte dimensioni dei segmenti, e' quasi invisibile sul video.

Rammentiamo, ora, la proprieta' evidenziata nella figura 5.22 per cui, quando esistono particolari rapporti tra i parametri della **window**, si producono delle figure in scala, tutte aventi in comune il vertice inferiore sinistro del video, cioe' il pixel di coordinate (0,0).

Prima di affrontare il problema della rappresentazione grafica di una funzione, esaminiamo gli effetti prodotti sulla rappresentazione degli assi (figura 7.2) muovendo le variabili di controllo del seguente programma **wincart4**.

```
10 'wincart4:finestre in scala
20 CLS:KEY OFF
30 INPUT"Risol. =",Z
35 IF Z=0 THEN 30
40 SCREEN Z
45 X=0:Y=0
50 INPUT"xmin = ",XMIN
60 INPUT"ymin = ",YMIN
70 INPUT"XMAX = ",XMAX
75 IF XMAX=XMIN THEN 70
80 INPUT"YMAX = ",YMAX
85 IF YMAX=YMIN THEN 80
90 WINDOW (XMIN,YMIN)-(XMAX,YMAX)
100 PSET (X,0)
110 X=X+.002
120 PSET (0,Y)
130 Y=Y+.002
135 IF X>1 OR Y>1 THEN 145
140 GOTO 100
145 PRINT"Battere (/) per ricominciare"
150 QQ$=INPUT$(1)
155 IF QQ$="/" THEN 30
160 GOTO 45
```

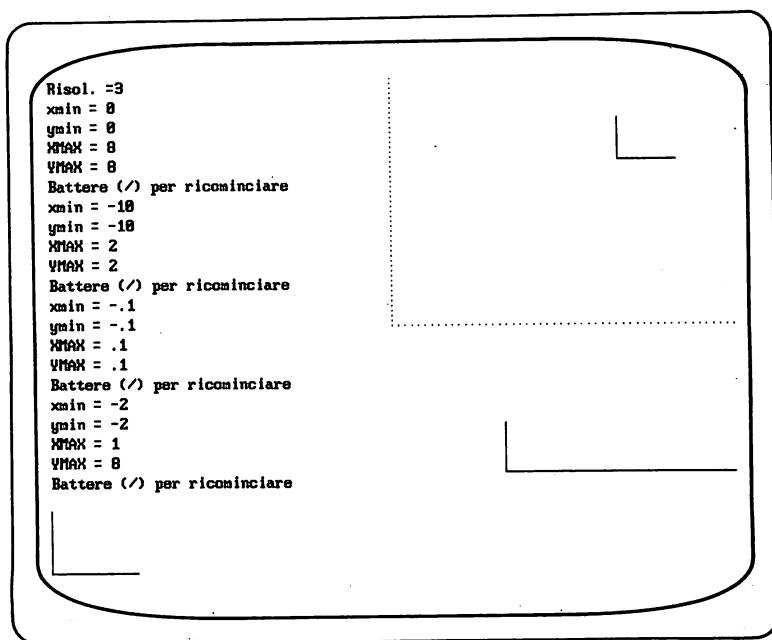



Figura 7.2

Si noti che la window alla linea 90 riporta il senso delle coordinate alle convenzioni cartesiane: la **x** crescente verso destra e la **y** crescente verso l'alto.

Quando i valori di xmin e ymin sono entrambi nulli, l'origine del nuovo riferimento coincide con il vertice sinistro in basso del video (schermo fisico). In tal caso (xmin=ymin=0) i valori assegnati a xmax e ymax definiscono il rapporto di scala tra i segmenti disegnati nel nuovo riferimento e quelli nel vecchio. Quando xmin=ymin=0, xmax=639 e ymax=399 il rapporto di scala e' 1:1. Tra l'altro, in tali condizioni, le istruzioni 100,110,120,130 disegnano appena tre punti, compreso quello nell'origine x=y=0.

Tra le varie passate la piu' istruttiva e' quella che genera la punteggiata (xmin=ymin=-0.1; xmax=ymax=0.1) per due motivi.

Il primo e' che la nuova origine sta nel centro dello schermo fisico; le coordinate del vertice in basso a sinistra dello schermo, rispetto a tale centro, sono appunto (-0.1,-0.1). Il secondo e' che la linea risulta una punteggiata molto rarefatta: cio' dipende dal fatto che l'incremento 0.002 assegnato ad ogni ciclo (goto alla linea 140), risulta amplificato dal rapporto di scala.

Passando al successivo programma **graf11**, notiamo che alla linea 70 i parametri della **window** soddisfano proprio alle condizioni prima indicate; pertanto, quando con la 30 assegniamo al **campo** il valore S=128, otteniamo lo stesso risultato gia' visto col programma **graf** prima descritto; qui pero', per la definizione interna dei parametri della window tutti legati ad S, sara' sempre con origine degli assi

nel centro del video. Le coordinate della vecchia origine rispetto al centro del video sono appunto: X_o = -256 e Y_o = -128.

```
10 'graf11: per ingrandire
20 CLS:KEY OFF
25 INPUT "RISOL =",Z
28 SCREEN Z
30 INPUT"campo= ",S
40 IF S=0 GOTO 30
50 WINDOW (-2*S,-2*S)-(S,S)
60 LINE(-5,0)-(5,0)
70 FOR I=-5 TO 5 STEP 1
80 LINE (I,0)-(I,.1)
90 NEXT
100 LINE(0,-2.5)-(0,2.5)
105 FOR I=-2 TO 2
107 LINE (0,I)-(.1,I):NEXT
110 LINE(0,1)-(1,0)
130 W$=INPUT$(1)
140 GOTO 20
```

Si notino i due cicli di generazione delle unita' di misura sugli assi alle linee 70 e 105. Per S=1 il segmento programmato alla linea 110 e' al suo massimo ingrandimento, senza uscire dal video. Segue ora il programma **graf21**.

```
10 'graf21:grafici di funzioni
20 CLS:KEY OFF
25 INPUT "RISOL.=",Z
27 SCREEN Z
30 INPUT"campo= ",S
40 INPUT"graf = ",G
50 INPUT"coeff= ",K
60 IF S=0 GOTO 30
65 PRINT
70 WINDOW (-1.6*S,-S)-(1.6*S,S)
75 LINE (-5,0)-(5,0)
80 FOR I=-5 TO 5
85 LINE (I,0)-(I,.1):NEXT
87 LINE (0,-2.5)-(0,2.5)
90 FOR I=-2 TO 2
95 LINE (0,I)-(.1,I):NEXT
100 LINE(0,1)-(1,0)
110 W$=INPUT$(1)
115 IF W$="/" THEN 20
120 FOR X=-G TO G STEP .1
130 Y= K*X ^2
140 LINE -(X,Y):NEXT
150 GOTO 30
```

In questo programma si traccia una parabola passante per l'origine e simmetrica rispetto all'asse delle ordinate la cui equazione e' scritta alla linea 130. Il tracciamento e' affidato alla **line** scritta al passo 140 che si appoggia come riferimento all'ultimo punto tracciato, cioe' il punto di coordinate (1,0) appartenente alla retta disegnata con la linea 100. Questo e' il motivo della linea spuria tracciata tra il punto (1,0) e l'estremita' della prima corda generata dalla FOR...NEXT al passo 120. I limiti inferiore e superiore della FOR sono imposti dal parametro G (graf) manovrato esternamente con la input scritta alla linea 40. Nelle figure 7.3 e 7.4 appaiono i risultati per alcune passate di graf21 nelle risoluzioni 2 e 3. Si notino gli effetti quando il coefficiente e' negativo e la scala passa dal valore S=3 a quello S=1.5.

Apportiamo ora alcune migliorie a graf2 introducendo la finestra dedicata alla grafica ed eliminando il segmento spurio prima segnalato: cio' avviene grazie a un'istruzione **preset** che, senza lasciar traccia, stabilisce un riferimento dinamico per la **line generatrice.**

Segue cosi' il nuovo programma **graf3** che ingloba le migliorie sopra indicate e grafica equazioni tipo y=k*X^N, con **k** e **N** parametri esterni. Anche qui si puo' scegliere il tipo di risoluzione grafica, la scala (campo), il dominio (graf) della variabile indipendente, il coefficiente e l'esponente. Nelle figure 7.5 (RIS=3) e 7.6 (RIS=1) vediamo i grafici di funzioni esponenziali ottenute variando il coefficiente (1,-1) e l'esponente (1, 3, 8, 12).

```
10 'graf3:grafici di funzioni
20 CLS:KEY OFF
25 INPUT "RISOL.=",Z
27 SCREEN Z:IF Z<2 THEN H=1 ELSE H=2
28 WIDTH 12*H
30 INPUT"campo= ",S
40 INPUT"graf = ",G
50 INPUT"coeff= ",K
55 INPUT "espon.=",N
56 IF N<1 THEN 55
57 IF G*INT(N)>15 THEN 55
58 IF G*K^INT(N)>511 THEN 55
60 IF S=0 GOTO 30
65 PRINT
66 IF Z<2 THEN A=1 ELSE A=2
67 IF Z<3 THEN B=1 ELSE B=2
70 WINDOW (-1.6*S,-S)-(1.6*S,S)
73 VIEW (100*A,35*B)-(312*A,167*B),,1
75 LINE (-5,0)-(5,0)'         ---->     asse X
80 FOR I=-5 TO 5    '         unita' di misura
85 LINE (I,0)-(I,.1):NEXT'    .....sull'asse X
87 LINE (0,-2.5)-(0,2.5)'     ---->     asse Y
90 FOR I=-2 TO 2    '         unita' di misura
95 LINE (0,I)-(.1,I):NEXT '        sull'asse Y
100 LINE(0,1)-(1,0)'     retta che intercetta
110 '           ...gli assi nei punti P=1; Q=1
```

```
111 N1=INT(N): D=(-1)^ N1*K*G^ N1
112 PRESET (-G,D)
120 FOR X=-G TO G STEP .05
130 Y= K*X ^N1
140 LINE -(X,Y):NEXT
145 W$=INPUT$(1)
148 IF W$="/" THEN 20 ELSE IF W$=" " THEN CLS
150 GOTO 30
```

Il programma e' sufficientemente protetto da errori derivanti da incompatibilita' tra i dati introdotti.

A tal scopo servono le linee 27, 66, 67 per adattare la finestra proiettiva alla risoluzione scelta. Le protezioni alle linee 57, 58 evitano dati in ingresso che farebbero assumere alla funzione valori inaccettabili per le istruzioni grafiche (preset e line).

Si notino infine le funzioni di governo del programma: [/] per tornare daccapo e [SPAZIO] per cancellare la finestra.

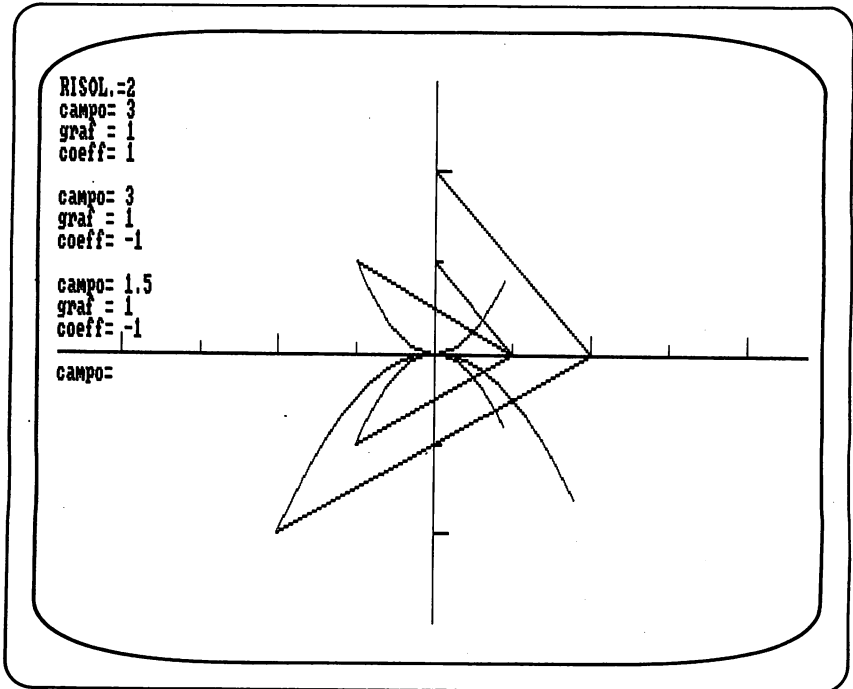



Figura 7.3

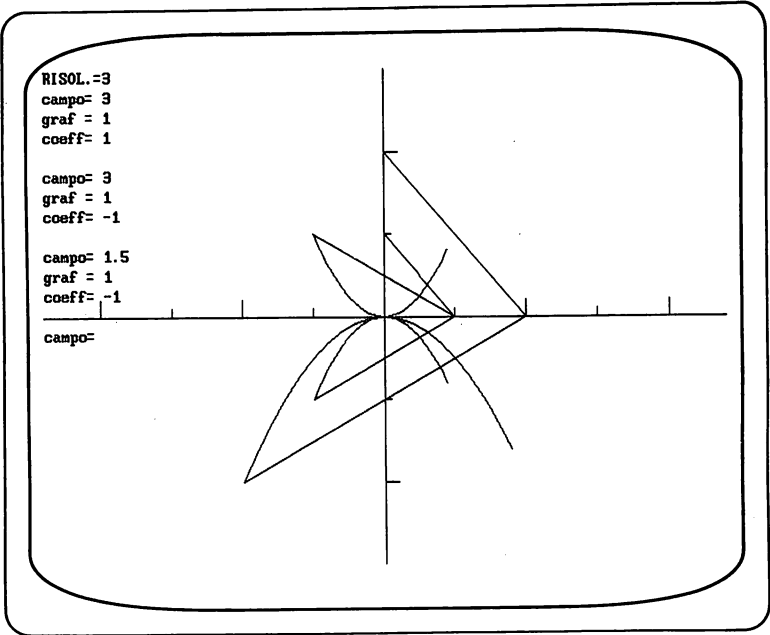



Figura 7.4

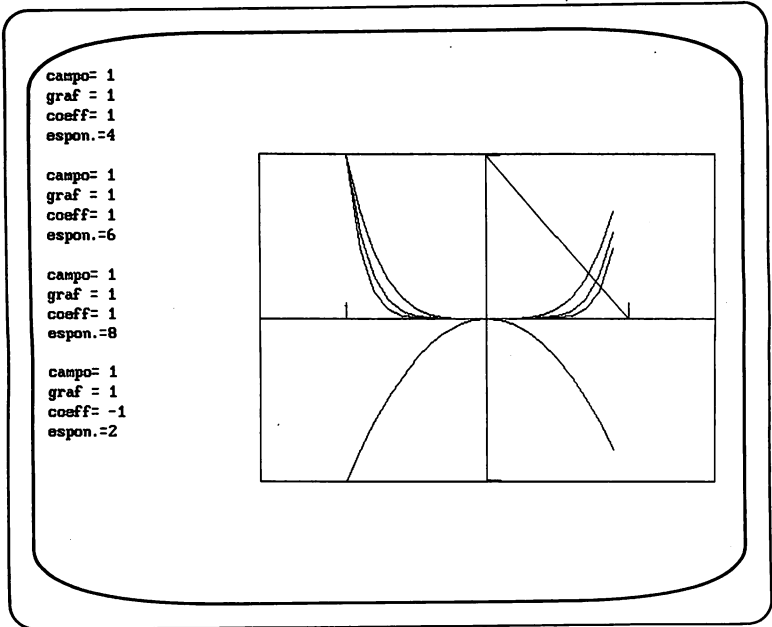



Figura 7.5

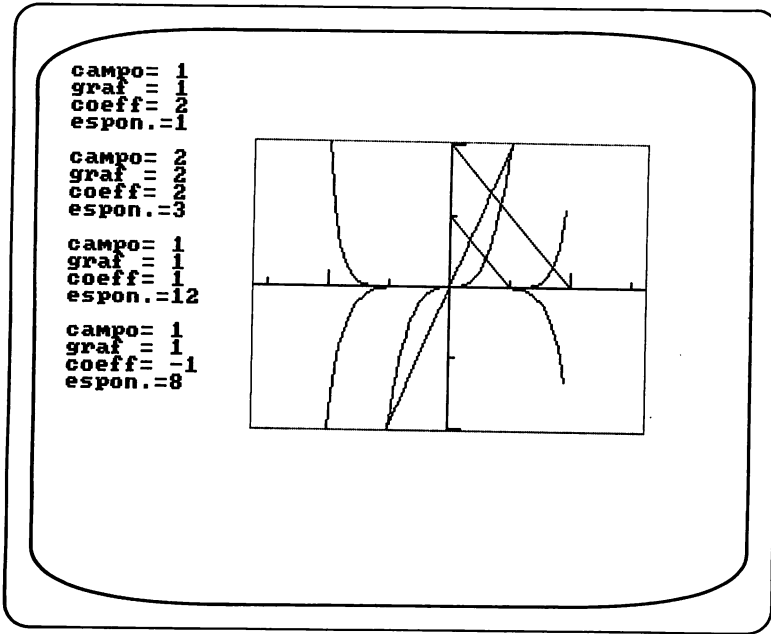



Figura 7.6

Una variante di graf3 e' il programma graf6. Nella figura 7.7 si sperimentano grafici di funzioni variando l'esponente, a parita' di scala e di coefficiente.

```
10 'graf6:grafici di funzioni
15 SCREEN 0,0,0
20 CLS:KEY OFF:INPUT "RISOL.=",Z
22 SCREEN Z
25 IF Z<2 THEN H=1 ELSE H=2
30 LOCATE 1,26:INPUT "campo=",S
32 IF S=0 GOTO 30
35 IF S=<0 THEN LOCATE 1,26:PRINT STRING$(6*H,32):GOTO 30
50 LOCATE 2,26:INPUT "coeff=",K
55 LOCATE 1,1:INPUT"espon.= ",N
56 IF N<0 THEN 55
57 '          --> Grafici di una funz. con esp. crescente
58 LOCATE 1,1:WIDTH 10*H
65 IF Z<2 THEN A=1 ELSE A=2
66 IF Z<3 THEN B=1 ELSE B=2
70 WINDOW (-1.6*S,-S)-(1.6*S,S)
73 VIEW (100*A,35*B)-(312*A,167*B),,1
75 LINE(-5,0)-(5,0)
80 FOR I=-5 TO 5' --->asse X
85 LINE (I,0)-(I,.1):NEXT
```

```
87 LINE(0,-2.5)-(0,2.5) '---->asse Y
90 FOR I=-2 TO 2
95 LINE (0,I)-(.1,I):NEXT
100 LINE(0,1)-(1,0)
112 PRINT "N="N,
115 D=(-1)^N*K
118 PRESET (-1,D)
120 FOR X=-1 TO 1 STEP .025
130 Y= K*X^N
140 LINE -(X,Y)
141 NEXT
148 W$=INPUT$(1):IF W$="/" THEN 20 ELSE IF W$=" " THEN CLS
150 N=N+1:GOTO 57
```

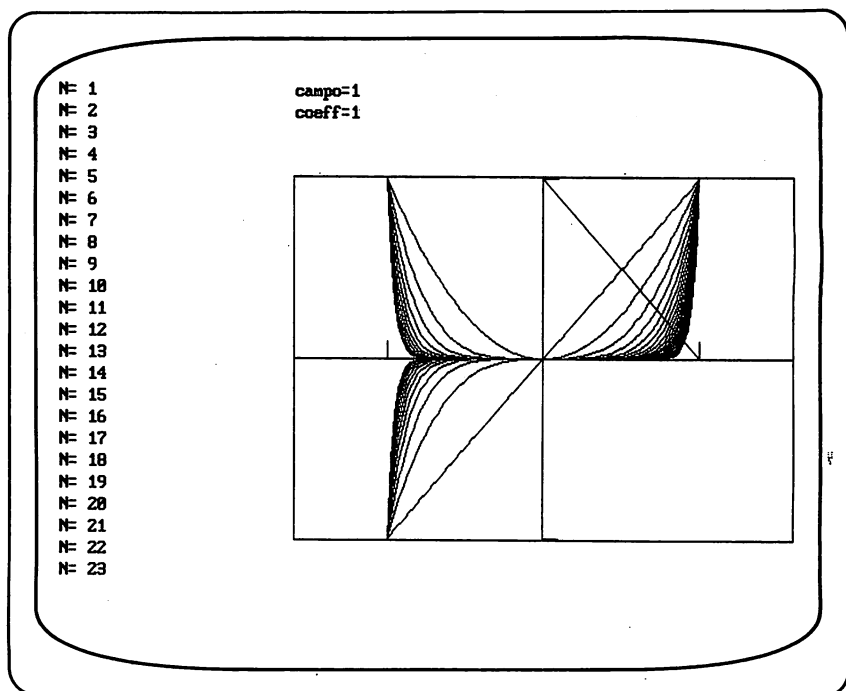



Figura 7.7

Seguono ora alcuni programmi per il tracciamento di funzioni trigonometriche. Il piu' elementare e' **senz**: traccia in screen 2 (media risoluzione 640x,200y) sinusoidi di ampiezza variabile.

```
10 'senz
20 KEY OFF:CLS:SCREEN 2
40 INPUT"amp.=",K
50 INPUT "PASSO=",P:IF P<1 THEN 50
60 FOR I=-300 TO 300 STEP P
62 X=3.141592*I/150
63 Y=K*SIN(X)
65 LINE-(I+300,100-Y*90)
100 NEXT:QQ$=INPUT$(1)
110 IF QQ$="=" THEN 20
120 GOTO 40
```

I diversi valori della funzione vengono calcolati mediante un ciclo con variabile di controllo I che, variando da -300 a +300, influenza corrispondentemente l'argomento x del seno calcolato alla linea 62. Qui, per I=-300, x vale all'incirca (-2*pigreco); con tale argomento il valore del seno e' zero. Altri valori di I, nel campo -300÷300 in cui il seno si annulla, sono I=-150, I=0, I=150 e I=300.

Il tracciamento avviene per piccoli segmenti individuati da una **line** inserita nello stesso ciclo ed espressa nella forma contratta: il primo estremo del segmento coincide quindi col secondo estremo del precedente segmento. Nella copia grafica, ottenuta in media risoluzione (figura 7.8), si noti la retta orizzontale dovuta al fatto che il segmento iniziale, non essendo preceduto da alcun segmento, assume come riferimento implicito il centro dello schermo, che in screen 2 ha coordinate 320,100.

In esecuzione, e' interessante scegliere passi di tracciamento diversi (che influenzano il numero dei punti in cui il ciclo calcola la funzione) in modo da vedere che a un passo piu' piccolo corrisponde un maggior numero di lati della spezzata che approssima la sinusoide.

La seguente variante del programma, denominata **senz1**, consente la scelta della risoluzione grafica (1=320x,200y; 3=640x,400y); inoltre, mediante la PRESET di linea 55 e' possibile assumere come primo riferimento della **line** proprio il primo valore della funzione seno. In figura 7.9 si riportano alcuni esempi di tracciamento.

```
10 'senz1
20 KEY OFF:CLS
30 INPUT "RISOL.=",Z:IF Z=0 THEN 30
32 IF Z<2 THEN R=2 ELSE R=1
33 IF Z<3 THEN Q=2 ELSE Q=1
35 SCREEN Z
40 LOCATE 2,1:PRINT "        "
45 LOCATE 2,1:INPUT"amp.=",K
47 LOCATE 3,1:PRINT "        "
48 LOCATE 3,1
50 INPUT "PASSO=",P:IF P<1 THEN 50
```

```
55 PRESET (0,200/Q)    'Riferimento della line
60 FOR I=-300/R TO 300/R STEP P
62 X=3.141592*I/150    'Argomento in radianti
63 Y=K*SIN(X)
65 LINE -(I+300/R,200/Q-Y*90/R)
66 NEXT:LOCATE 1,1
67 PRINT"<CR>:Ancora <=>:Daccapo"
100 QQ$=INPUT$(1)
110 IF QQ$="=" THEN 20
120 GOTO 40
```

Segue, ora, il programma **insenz1** per il tracciamento della funzione y=sen(1/x).

Il programma e' stato ricavato dallo schema di senz1 con la sola modifica di un filtro del valore x=0 (linea 61) senza il quale si provocherebbe un blocco del programma per **division by zero**: infatti quando la variabile di controllo I assumesse il valore zero, per la linea 62, si annullerebbe anche la x e l'argomento del seno diventerebbe indeterminato. Un esempio e' offerto dalla figura 7.10.

Il programma puo' essere considerato un piccolo laboratorio per lo studio di qualsiasi altra funzione e non solo trigonometrica: basta sostituire, infatti, alla linea 63 la funzione da studiare.

```
10 'insenz1
20 KEY OFF:CLS
30 INPUT "RISOL.=",Z:IF Z=0 THEN 30
32 IF Z<2 THEN R=2 ELSE R=1
33 IF Z<3 THEN Q=2 ELSE Q=1
35 SCREEN Z
40 LOCATE 2,1:PRINT "          "
45 LOCATE 2,1:INPUT"amp.=",K
47 LOCATE 3,1:PRINT "          "
48 LOCATE 3,1
50 INPUT "PASSO=",P:IF P<1 THEN 50
55 PRESET (0,200/Q)    'Riferimento della line
60 FOR I=-300/R TO 300/R STEP P
61 IF I=0 THEN 66
62 X=3.141592*I/150    'Argomento in radianti
63 Y=K*COS(X)
65 LINE -(I+300/R,200/Q-Y*90/R)
66 NEXT:LOCATE 1,1
67 PRINT"<CR>:Ancora <=>:Daccapo"
100 QQ$=INPUT$(1)
110 IF QQ$="=" THEN 20
120 GOTO 40
```

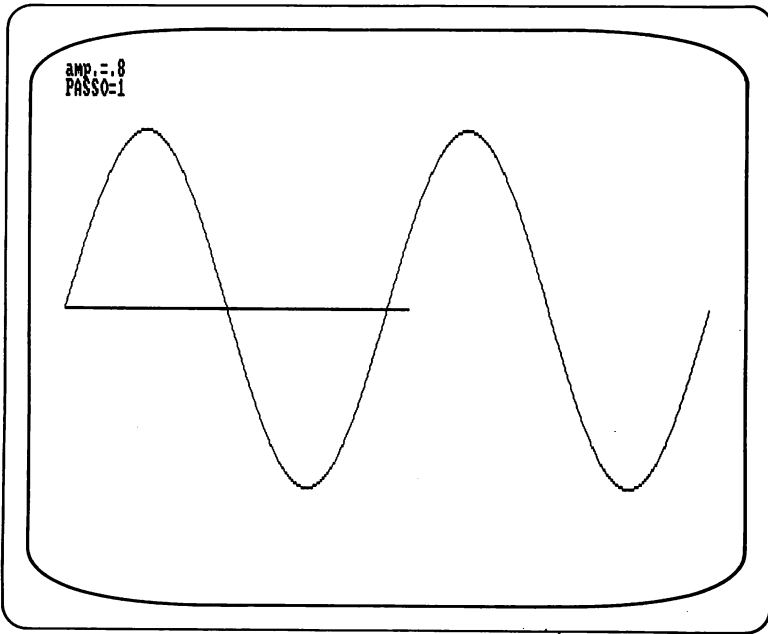



Figura 7.8

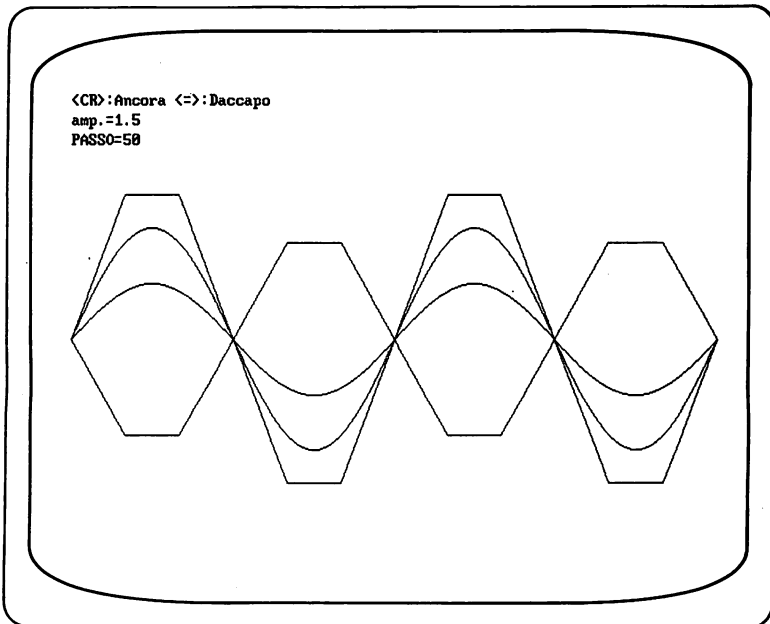



Figura 7.9

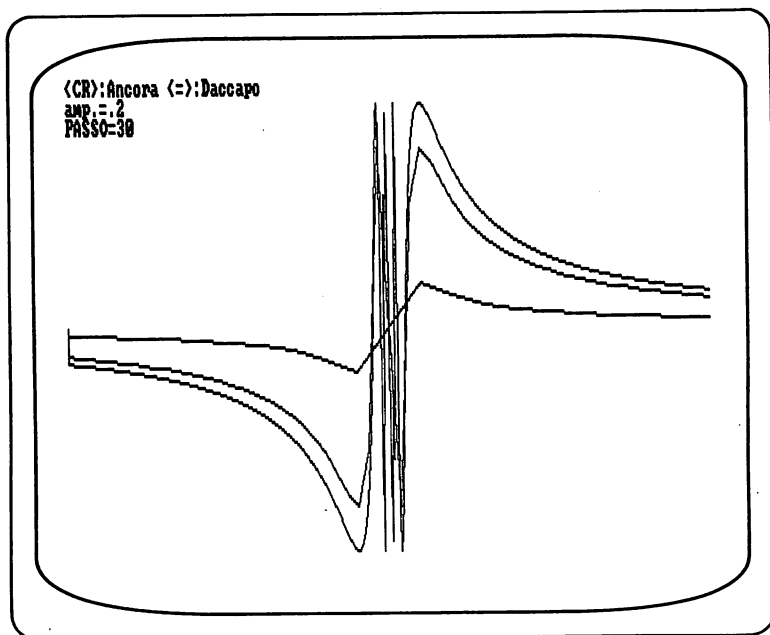



Figura 7.10

Un approccio diverso di analisi e di programmazione e' offerto dal seguente programma **persen2** con il quale si possono tracciare sinusoidi scegliendone il periodo e l'ampiezza in un quadro di riferimento che ne facilita l'interpretazione (figura 7.11).

```
10 'persen2
20 KEY OFF:CLS
30 INPUT "RISOL.=",Z:IF Z=0 THEN 30
32 IF Z<2 THEN R=2 ELSE R=1
33 IF Z<3 THEN Q=2 ELSE Q=1
35 SCREEN Z:GOSUB 1020
40 LOCATE 3,1:PRINT "        "
45 LOCATE 3,1:INPUT"per.=",D
47 LOCATE 4,1:PRINT "        "
48 LOCATE 4,1
50 INPUT "AMP.=",K:IF K=0 THEN 50
55 PRESET (0,200/Q)     'Riferimento della line
58 F=319/D
60 FOR I=0 TO 640/R STEP 2
62 X=3.141592*I/F       'Argomento in radianti
63 Y=-K*SIN(X)+200
65 LINE -(I,Y/Q)
66 NEXT:LOCATE 1,1
```

```
67 PRINT"CR:Ancora"
68 PRINT"d:Daccapo"
100 QQ$=INPUT$(1)
110 IF QQ$="d" THEN 20
120 GOTO 40
1020 LINE (0,199/Q)-(639/R,199/Q)
1030 LINE (319/R,399/Q)-(319/R,0)
1040 LINE (0,300/Q)-(638/R,99/Q),,B
1050 LINE (160/R,398/Q)-(479/R,0),,B
1055 RETURN
```

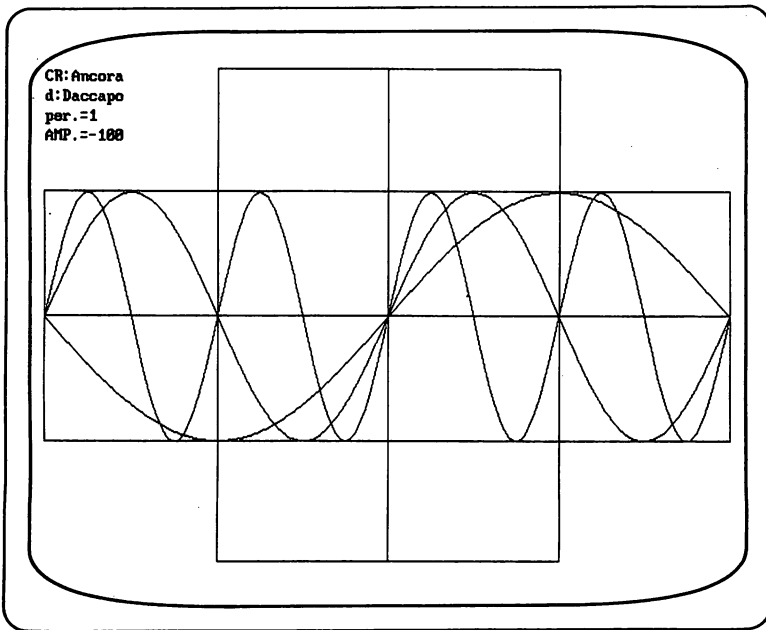



Figura 7.11

Sul tracciamento delle curve potremmo scrivere un libro a se'. Proponiamoci, a questo punto, di materializzare gli elementi di un vettore numerico, disegnando su video un **istogramma** o diagramma a barre. I parametri per il tracciamento sono, quindi, il numero degli elementi e il loro valore che, graficamente, verranno tradotti in numero di barre con le rispettive altezze.

Come impostazione iniziale puo' servire quel programma **retcom** descritto al paragrafo 5.1: tale programma genera un numero di rettangoli o barre di base (B) e altezza (H) date, a distanza P costante l'uno dall'altro.

Se nell'espressione della line al passo 80 sostituiamo l'altezza H con il termine generico V(E) di una variabile con indice, tale **line** generera' una barra di altezza pari al valore dell'elemento E considerato. Chiaramente, dopo aver fatto coincidere la variabile di controllo del ciclo con l'indice del vettore, la **line** diventera':

```
line(10+E*P,0)-(10+B+E*P,50+V(E)),,BF
```

Segue il programma **liret1** che, dopo aver caricato un vettore di N elementi introdotti da tastiera, li assegna a un ciclo nel corpo del quale c'e' una **line** generatrice di barre. Questa line, pero', e' diversa da quella che compare nel programma retcom.

La differenza risiede nell'aver reso "elastiche" le misure della base e del passo orizzontale, cosi' da poter far stare nel video anche istogrammi con un elevato numero di elementi: per far questo i termini B (base) e P (passo) sono stati rapportati a N.

```
5 'liret1: istogrammi da una lista di numeri pseudocasuali
10 CLS:KEY OFF:SCREEN 2:CLEAR
15 WINDOW (0,0)-(639,319)
20 INPUT "n.elem.=",N
30 DIM V(N)
40 T$=MID$(TIME$,7)
50 T=VAL(T$)
60 RANDOMIZE(T)
70 INPUT;"  da <0> a < ",L:PRINT">"
80 FOR E=1 TO N
90 R=FIX((L+1)*RND):V(E)=R:PRINT V(E)" ";:NEXT E
100 PRINT
110 PRINT
170 INPUT "passo orizzont.=",P
180 INPUT;"base istogr.=",B
190 FOR E=1 TO N
200  LINE (10+(E-1)*P/N,10)-(10+B/N+(E-1)*P/N,10+V(E)),,B
210 NEXT E
220 A$=INPUT$(1)
230 GOTO 10
```

Il disegno di figura 7.12 illustra la nuova situazione insieme a un istogramma ottenuto con una passata di liret1; in figura 7.13 la versione **liret2** del programma che traccia istogrammi in finestra dedicata.

30
20
10
0
P/N
P/N
P/N
P/N
(X1 , 10+V(E))
(10,10)
E=1
E=2
E=3
E=4
E=5
10
10
50
100
10
B/N

n.elem.=10
da <0> a < 200>
118 0 93 35 41 58 90 191 185 46
passo orizzont.=500
base istogr.=300

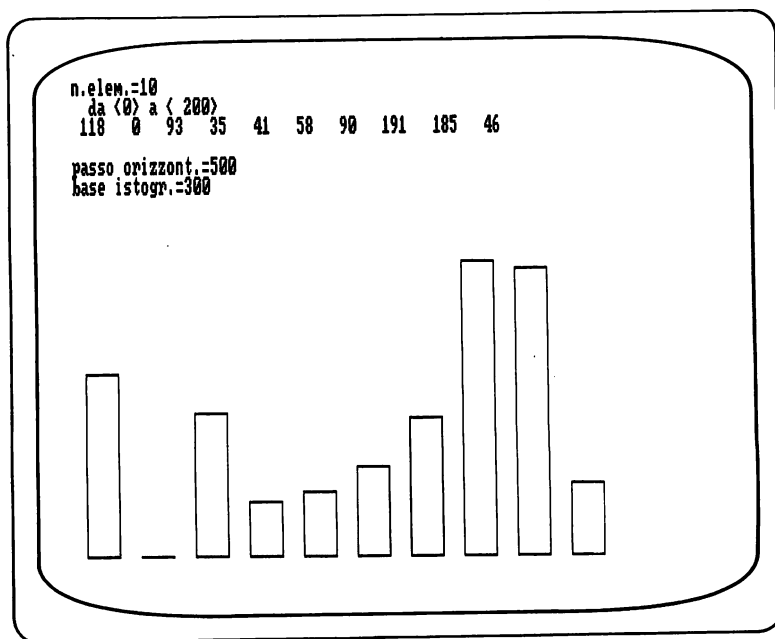
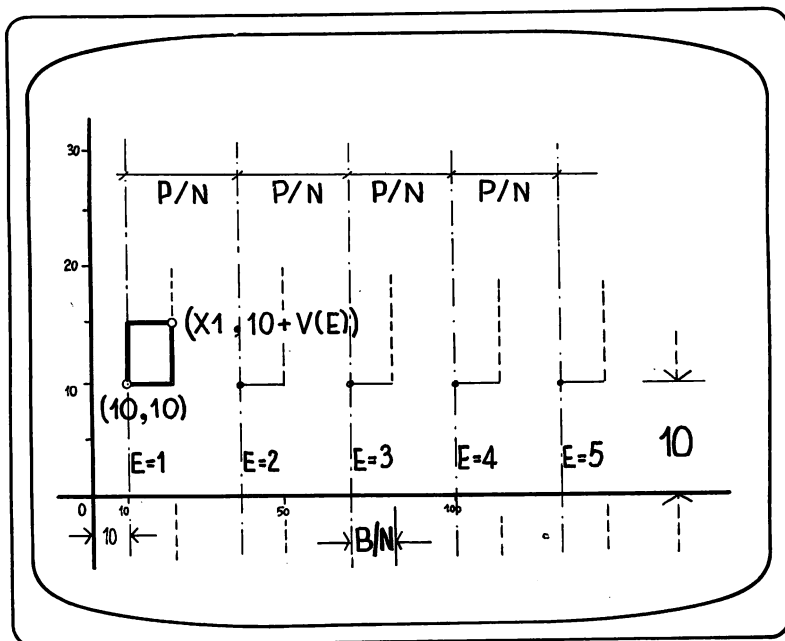



Figura 7.12

```
5 'liret2: istogrammi da una lista di numeri pseudocasuali
10 CLS:KEY OFF:SCREEN 3:CLEAR
12 WIDTH 20
15 WINDOW (0,0)-(639,319)
17 VIEW (200,20)-(638,398),,1
20 INPUT "n.elem.=",N
30 DIM V(N)
40 T$=MID$(TIME$,7)
50 T=VAL(T$)
60 RANDOMIZE(T)
70 INPUT;"  da <0> a < ",L:PRINT">"
80 FOR E=1 TO N
90 R=FIX((L+1)*RND):V(E)=R:PRINT V(E)" ";:NEXT E
100 PRINT
110 PRINT
170 INPUT "passo orizzont.=",P
180 INPUT;"base istogr.=",B
190 FOR E=1 TO N
200  LINE (10+(E-1)*P/N,0)-(10+B/N+(E-1)*P/N,V(E)),,B
210 NEXT E
220 A$=INPUT$(1)
230 GOTO 10
```

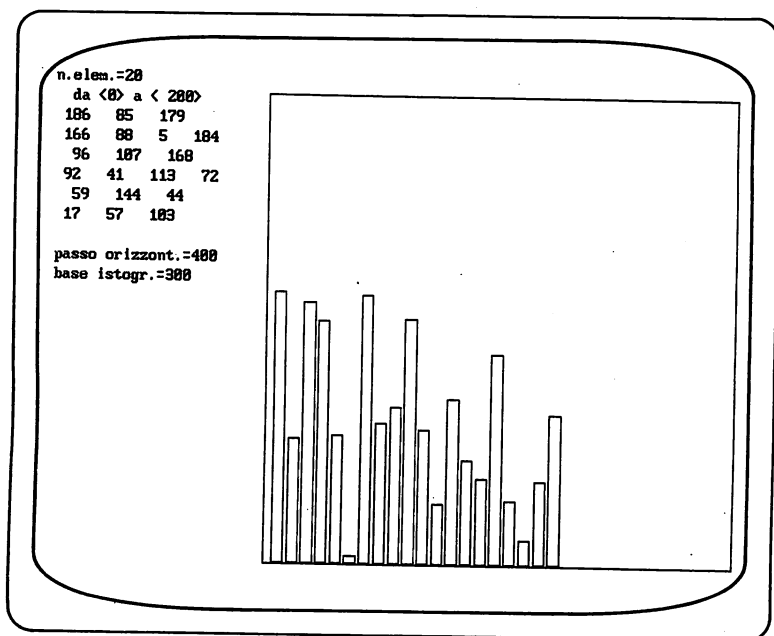



Figura 7.13

7.2 COME DISEGNARE SU VIDEO

Questa prestazione di macchina sembra superflua dopo tutto quanto abbiamo gia' scoperto fin qui in tema di grafica su M24. Ci riferiamo alle istruzioni elementari tipo **pset** e a quelle piu' sofisticate come la **line** e la **circle.** Che altro aggiungere?

L'istruzione **draw** (in inglese, disegnare) in effetti consente di realizzare dei disegni su video che potremmo fare anche con le istruzioni gia' note. Ma la sintassi e la logica della draw, come vedremo, permettono una programmazione piu' semplice per l'utente e, nello stesso tempo, piu' efficiente per disegni complessi. Le scelte che possono essere fatte tramite i sottocomandi della **draw** sono i semi di un vero e proprio linguaggio grafico.

Conviene imparare l'istruzione draw pensando a un'ipotetica penna, detta "penna virtuale", che possa scrivere in qualunque punto del video all'interno di una finestra. Gli elementi della sua sintassi sono numerosi e, con riferimento alla figura 7.14, inizieremo dai piu' essenziali.

Supponiamo, ad esempio, di essere in massima risoluzione grafica (640,400) e di voler disegnare un segmento di retta dal punto F (290,200) al punto G(340,300). Con la draw e' possibile far cio' in vari modi. Il primo consiste nel fissare un punto di riferimento con una **pset** nel punto F e poi scrivere una draw. Le istruzioni corrispondenti sono:

```
10  PSET (290,200)

20  DRAW  "M +50,75"
```

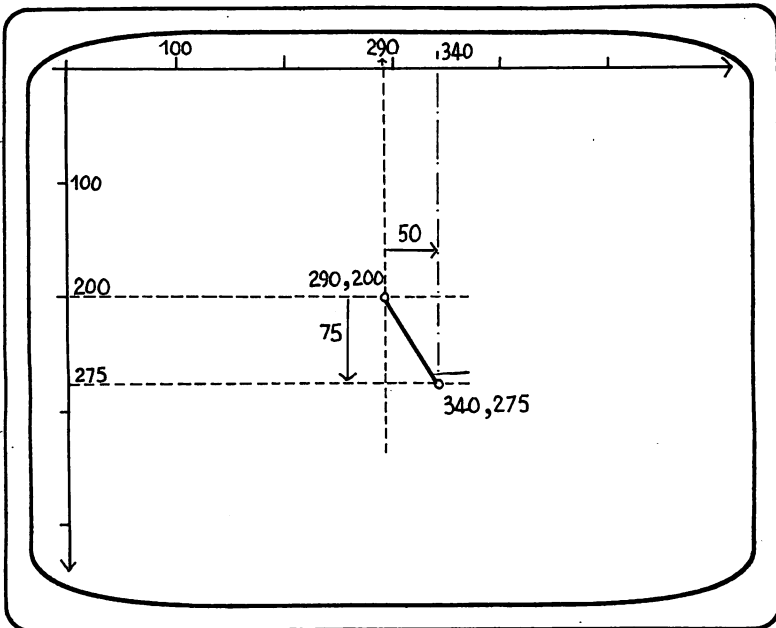



Figura 7.14

Tale **draw** esprime lo spostamento della penna in termini di incremento rispetto al punto di riferimento. Nella draw, pertanto, la lettera **M** significa: "Muovi" la penna dal punto di riferimento -stabilito con la pset (290,200)- al nuovo punto (340,275) ottenuto incrementando l'ascissa e l'ordinata del punto F, rispettivamente, di **50** e di **75.**
Si noti che il corpo della draw e' racchiuso da virgolette, consueta notazione per delimitare una stringa.

Un altro modo di ottenere lo stesso risultato e' quello di assegnare all'argomento della draw le coordinate assolute del secondo estremo del segmento da disegnare; si dovra' scrivere pertanto:

```
10  PSET (290,200)

20  DRAW  "M 340,275"
```

Un terzo modo per tracciare un segmento e' quello di spostare la "penna virtuale", dandogli un comando di spostamento orizzontale a **destra** (R) o a **sinistra** (L), in **basso** (D), in **alto** (U) e nei quattro sensi diagonali (E=alto destra), (F=basso destra), (G=basso sinistra) e (H=alto sinistra). Inoltre e' possibile lo spostamento a penna sollevata, che si ottiene premettendo alla sigla del comando la lettera **B** (blind=**cieco**). Con questi otto sottocomandi (16 se consideriamo anche gli spostamenti a penna sollevata, BR, BL, BD, etc.), si possono assegnare spostamenti funzionali indiretti attraverso i nomi di opportune variabili. Tali variabili devono essere precedute dal segno [=] e seguite dal segno [;]. Un esempio e' offerto dalle seguenti istruzioni:

```
282 PSET (400,300)
285 AA=30:BB=50
287 DRAW "h=aa;u=bb;"
```

Tra i comandi piu' potenti dell'istruzione **draw** esiste quello per eseguire una sottostringa: cio' consente, come GOSUB per i sottoprogrammi, di eseguire una catena di stringhe, ciascuna contenente le proprie specifiche grafiche. Ad esempio, assegnando a quattro variabili stringa (A$, B$, C$, D$,) i corrispondenti sottocomandi per generare i quattro lati di un rombo e concatenando queste quattro variabili in modo che globalmente siano a loro volta assegnate alla variabile ROM$ bastera' eseguire l'istruzione DRAW ROM$ per ottenere il disegno del rombo. In tal modo abbiamo costruito una procedura grafica che possiamo ripetere in qualsiasi momento semplicemente richiamandone il nome. Nella figura 7.15, riportiamo una copia grafica del disegno ottenibile con le istruzioni ivi listate.

Tra esse merita una menzione particolare quella alla linea 372 che contiene il sottocomando P. Tale opzione serve a colorare un poligono che viene individuato dalle coordinate dell'ultimo punto disegnato.

Nel caso particolare della figura, volendo interessare l'interno del secondo rombo e' stato necessario scrivere prima l'istruzione alla linea 370 che sposta, senza scrivere, la penna virtuale in un punto interno al rombo. Nel disegno appaiono anche delle cornici graduate, che ne facilitano la lettura. Tali riferimenti sono generati dalla prima parte del programma **giop** qui riportato.

```
10 'giop
20 CLS:KEY OFF:SCREEN 3
30 WIDTH 33
40 '..................asse X
50 LINE (0,399)-(639,399)
60 FOR J=0 TO 639 STEP 10
70 LINE (J,395)-(J,399):NEXT
80 FOR J=0 TO 639 STEP 100
90 LINE (J,390)-(J,399):NEXT
100 '..................asse Y
110 LINE (639,0)-(639,399)
120 FOR J=0 TO 399 STEP 10
130 LINE (635,J)-(639,J):NEXT
140 FOR J=0 TO 639 STEP 100
150 LINE (630,J)-(639,J):NEXT
160 '(X=270) PARALL....ASSE Y
170 LINE (270,0)-(270,300)
180 FOR J=0 TO 300 STEP 10
190 LINE (270,J)-(275,J):NEXT
200 FOR J=0 TO 300 STEP 100
210 LINE (270,J)-(279,J):NEXT
260 PSET (290,200)
```

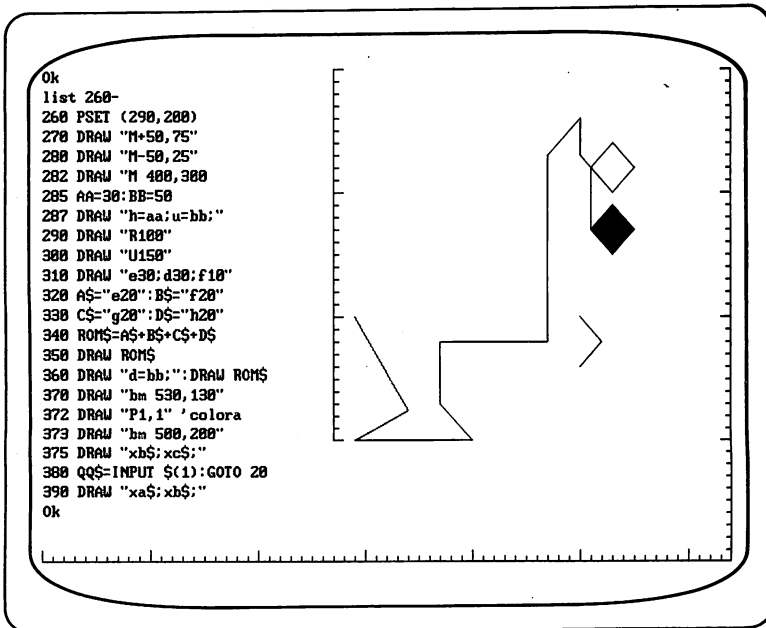



Figura 7.15

Un programma istruttivo che mostra come rendere variabile anche l'argomento della sottostringa e' **rom**, la cui lista appare nella copia grafica di alcune passate (figura 7.16). La variabile stringa introdotta con la input alla linea 30 e' l'ampiezza del lato del rombo; tale ampiezza viene "stringata" (concatenata) alle lettere e,f,g,h -che sono i nomi dei sottocomandi per la generazione di segmenti in diagonale- per formare le sottostringhe A$, B$, C$, D$, le quali, a loro volta, fuse nella variabile ROM$ costituiscono l'argomento della DRAW.

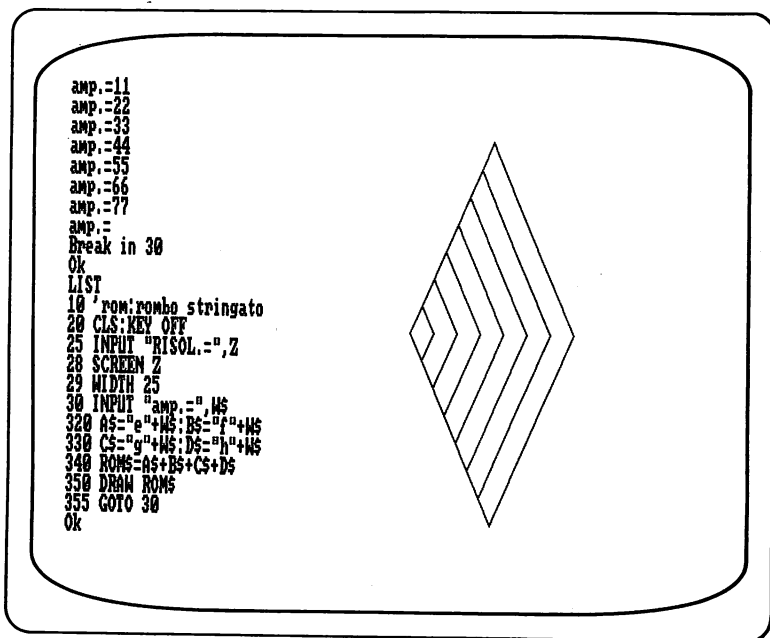



Figura 7.16

Vediamo ora un piccolo programma per disegnare delle figure che hanno al loro interno dei rapporti angolari. Ci serviremo ancora dello stesso programma **giop**, la cui variante denominata **gio** genera il disegno in figura 7.17. L'istruzione che contiene la nozione di angolo e' un'ulteriore scelta offerta da **draw**. Le specifiche di tale opzione sono le seguenti:

DRAW "ta**α**;nu**R**"	dove **α** e' l'angolo di tracciamento di un segmento rispetto alla direzione della Y, e **R** la lunghezza del segmento stesso.

Il programma **gio** offre anche un esempio di generazione di stelle mediante i sottocomandi **ta** e **nu** i cui argomenti variabili appartengono a una **draw** piazzata nel corpo di una FOR...NEXT. La variabile angolo assume i valori della variabile di controllo del ciclo che avanza fino a 360 gradi col passo **p** selezionabile dall'esterno con la INPUT alla linea 160. L'ampiezza del raggio **g** e' costante ma anch'esso selezionabile ogni volta che si fa girare il programma.

Segue il listato iniziale del programma **gio** (le linee dalla 280 alla fine sono contenute nella figura 7.17).

```
10 'gio
20 CLS:KEY OFF:SCREEN 3
30 WIDTH 33
40 '(Y=399) Parall....asse X
50 LINE (0,399)-(639,399)
60 FOR J=0 TO 639 STEP 10
70 LINE (J,395)-(J,399):NEXT
80 FOR J=0 TO 639 STEP 100
90 LINE (J,390)-(J,399):NEXT
100 '     ...........asse X
110 LINE (300,0)-(600,0)
120 FOR J=300 TO 600 STEP 10
130 LINE (J,0)-(J,5):NEXT
140 FOR J=300 TO 600 STEP 100
150 LINE (J,0)-(J,9):NEXT
160 '.................asse Y
170 LINE (639,0)-(639,399)
180 FOR J=0 TO 399 STEP 10
190 LINE (635,J)-(639,J):NEXT
200 FOR J=0 TO 639 STEP 100
210 LINE (630,J)-(639,J):NEXT
220 '(X=270) PARALL....ASSE Y
230 LINE (270,0)-(270,300)
240 FOR J=0 TO 300 STEP 10
250 LINE (270,J)-(275,J):NEXT
260 FOR J=0 TO 300 STEP 100
270 LINE (270,J)-(279,J):NEXT
280 INPUT "PASSO STELLA=",P
```

Segue ora la versione **gio1** che contiene alcuni esempi di concatenamento funzionale (linee 370 e 390) i cui effetti sono valutabili dalla copia riportata nella figura 7.18. La parte iniziale di **gio1** e' identica a quella di **gio**, la restante parte e' nella stessa figura 7.18

Nella versione **gio2** si esperimenta l'ulteriore opzione di **draw [DRAW S]** che riguarda la trasformazione apportata al disegno generato dalle draw che la seguono.

L'effetto scala introdotto dipende dal numero posto alla destra della lettera S e che puo' assumere valori da 1 a 255. Come e' d'obbligo in questo libro, dove ogni istruzione viene forzata al limite della parametricita', anche la draw S e' stata resa variabile per valori introdotti esternamente. La sintassi rispetta la regola del segno [=] prima della variabile e del [;] dopo la variabile esterna. Cosi' appare costituita la draw alla linea 325, dove la variabile **l** e' associata alla S. Il fattore di scala viene sempre diviso per quattro: cosi', quando si sceglie il valore 4, non si ha effetto scala. La massima riduzione si ottiene con il valore L=1.

Nel listato di gio2 -che riportiamo di seguito- notiamo alla linea 305 l'istruzione input per la definizione della scala.

```
10 'gio2
20 CLS:KEY OFF:SCREEN 3
30 WIDTH 33
40 '(Y=399) Parall....asse X
50 LINE (0,399)-(639,399)
60 FOR J=0 TO 639 STEP 10
70 LINE (J,395)-(J,399):NEXT
80 FOR J=0 TO 639 STEP 100
90 LINE (J,390)-(J,399):NEXT
100 '     ...........asse X
110 LINE (300,0)-(600,0)
120 FOR J=300 TO 600 STEP 10
130 LINE (J,0)-(J,5):NEXT
```

```
140 FOR J=300 TO 600 STEP 100
150 LINE (J,0)-(J,9):NEXT
160 '.................asse Y
170 LINE (639,0)-(639,399)
180 FOR J=0 TO 399 STEP 10
190 LINE (635,J)-(639,J):NEXT
200 FOR J=0 TO 639 STEP 100
210 LINE (630,J)-(639,J):NEXT
220 '(X=270) PARALL....ASSE Y
230 LINE (270,0)-(270,300)
240 FOR J=0 TO 300 STEP 10
250 LINE (270,J)-(275,J):NEXT
260 FOR J=0 TO 300 STEP 100
270 LINE (270,J)-(279,J):NEXT
280 INPUT "PASSO STELLA=",P
290 IF P=0 THEN 280
300 INPUT "RAGGIO STELLA=",G
305 INPUT "SCALA=",L
310 IF G=0 THEN 300
320 PSET (290,200)
325 DRAW "S=1;"
330 DRAW "M+50,75"
340 DRAW "M-50,25"
350 DRAW "R100"
360 DRAW "U150"
370 DRAW "e=g;d30;f=g;"
380 GOSUB 900
385 Z=2*G
390 DRAW "u=z;"
400 GOSUB 900
410 DRAW "L150"
420 DRAW "ta20;nu60"
430 DRAW "ta140;nu30"
440 QQ$=INPUT $(1):GOTO 20
900 FOR D=0 TO 360 STEP P
910 DRAW "TA=D;NU=g;"
920 NEXT D
930 RETURN
```

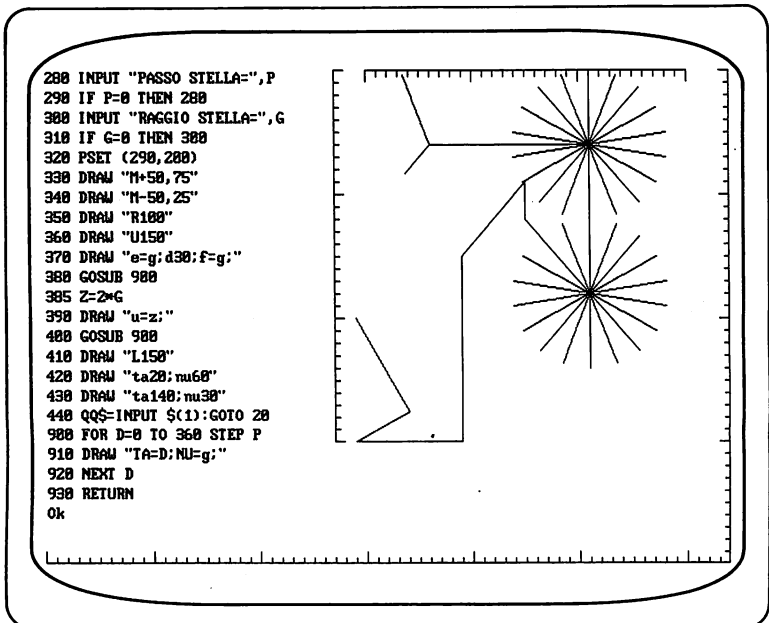



Figura 7.17

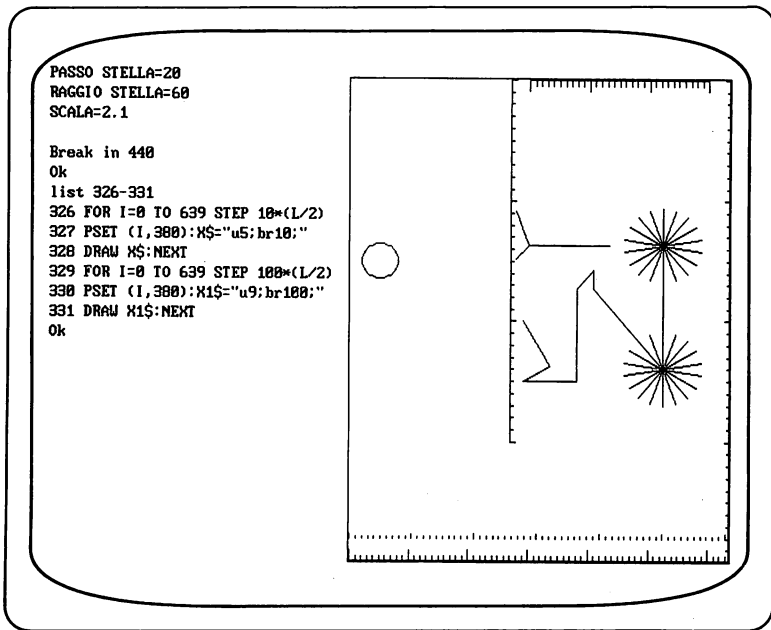



Figura 7.18

Per verificare l'influenza delle istruzioni **window** e **view** sui disegni generati dalla draw si possono fare esperimenti con il programma giof31, e osservare:

- l'effetto scala (1:1) imposto dalla window (linea 35) sulla finestra proiettiva influenza solo le primitive grafiche diverse dalla draw: infatti, variando la scala della draw alla linea 325, il righello (disegnato con le istruzioni listate nella stessa figura 7.19) cambia rapporto rispetto a quello ottenuto sul contorno della finestra, mentre il cerchio della linea 331 non viene alterato;

- la precisione nelle diverse risoluzioni grafiche;

- come sistemare i parametri nelle diverse istruzioni per tener conto della risoluzione grafica (screen 1,2,3) e non violare i limiti in ciascuna di esse (linee 28 e 29).

Segue il programma giof31.

```
10 'giof31
20 CLS:KEY OFF
25 INPUT"RIS.=",Z
26 IF Z=0 THEN 25
27 SCREEN Z
28 IF Z=1 THEN K=1 ELSE K=2
29 IF Z<3 THEN Y=1 ELSE Y=2
30 WIDTH 33
35 WINDOW SCREEN (0,0)-(639,399)
37 VIEW (140*K,10)-(318*K,199*Y),,1
40 '(Y=399) Parall....asse X
50 LINE (0,399)-(639,399)
60 FOR J=0 TO 639 STEP 10
70 LINE (J,395)-(J,399):NEXT
80 FOR J=0 TO 639 STEP 100
90 LINE (J,390)-(J,399):NEXT
100 '         ...........asse X
110 LINE (300,0)-(600,0)
120 FOR J=300 TO 600 STEP 10
130 LINE (J,0)-(J,5):NEXT
140 FOR J=300 TO 600 STEP 100
150 LINE (J,0)-(J,9):NEXT
160 '..................asse Y
170 LINE (639,0)-(639,399)
180 FOR J=0 TO 399 STEP 10
190 LINE (635,J)-(639,J):NEXT
200 FOR J=0 TO 399 STEP 100
210 LINE (630,J)-(639,J):NEXT
220 '(X=270) PARALL....ASSE Y
230 LINE (270,0)-(270,300)
240 FOR J=0 TO 300 STEP 10
250 LINE (270,J)-(275,J):NEXT
260 FOR J=0 TO 300 STEP 100
270 LINE (270,J)-(279,J):NEXT
280 INPUT "PASSO STELLA=",P
290 IF P=0 THEN 280
300 INPUT "RAGGIO STELLA=",G
301 INPUT "SCALA=",L
310 IF G=0 THEN 300
320 PSET (290,200)
325 DRAW "S=1;"
326 FOR I=0 TO 639 STEP 10*(L/2)
327 PSET (I,380):X$="u5;br10;"
328 DRAW X$:NEXT
329 FOR I=0 TO 639 STEP 100*(L/2)
330 PSET (I,380):X1$="u9;br100;"
331 DRAW X1$:NEXT
332 CIRCLE (50,150),30
334 PSET (290,200)
335 DRAW "M+50,75"
340 DRAW "M-50,25"
350 DRAW "R100"
```

```
360 DRAW "U150"
370 DRAW "e30;d30;f130"
380 GOSUB 900
385 Z=2*G
390 DRAW "u200"
400 GOSUB 900
405 DRAW "bl100"
410 DRAW "L150"
420 DRAW "ta20;nu60"
430 DRAW "ta140;nu30"
440 QQ$=INPUT $(1):GOTO 20
900 FOR D=0 TO 360 STEP P'..stella
910 DRAW "TA=D;NU=g;"
920 NEXT D
930 RETURN
```

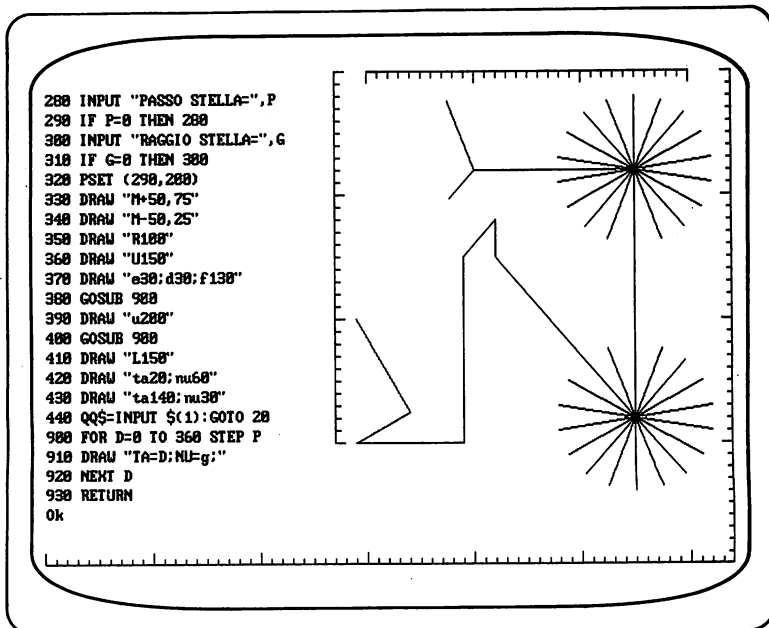



Figura 7.19

7.3 IL TRASFERIMENTO DELLE IMMAGINI

Per trasportare un disegno o una porzione dei pixel accesi da una zona dello spazio grafico del video a un'altra, il repertorio di GW-BASIC mette a disposizione due potenti istruzioni: la **GET** e la **PUT**.

Con la GET si fa la scansione della porzione interessata, i cui attributi grafici vengono immagazzinati in una opportuna matrice in memoria; con la PUT tali attributi vengono restituiti in un'altra zona.

Il trasporto avviene, quindi, in modo automatico senza alcuna necessita' di ridisegnare la figura. Descriviamo ora il dettaglio delle operazioni. Innanzitutto, occorre definire nella GET la zona di cui si vuole trasportare altrove l'immagine video. Le modalita' per indicarla sono le coordinate dei vertici di un rettangolo che circoscrive la zona, come faremmo nell'istruzione **line** (figura 7.20).

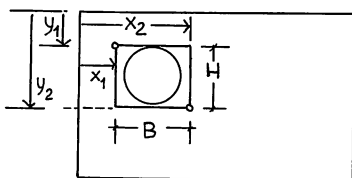



Figura 7.20

Ma prima di scrivere la GET in un programma, bisogna definire le dimensioni della matrice che memorizza gli attributi grafici della zona. Abbiamo calcolato per le tre risoluzioni grafiche disponibili su M24 una formula generale che consente di calcolare le dimensioni DZ della zona:

$$DZ = 4 + H\ INT \frac{(P.B + 7)}{8}$$

dove H e B sono l'altezza e la base del rettangolo e P vale 2 per lo screen 1 e 1 per lo screen 2 e 3.

Siamo, quindi, in grado di usare la GET, la cui forma generale e':

```
GET (X1,Y1)-(X2,Y2),NZ
```

dove X1, Y1, e X2,Y2 sono le coordinate dei vertici del rettangolo che incornicia la zona (figura 7.20) e NZ il nome della matrice.

Il programma **dispo**, per spostare un disegno in screen 1, incomincia con le istruzioni che creano il disegno da spostare. Sia, ad esempio, un cerchio con centro in (100,100) di raggio 20 (linea 30).

```
2 'dispo:spostare un disegno
5 CLS:KEY OFF
10 SCREEN 1
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((2*H+7)/8)
50 DIM NZ(DZ)
60 GET (80,80)-(120,120),NZ
70 PUT (200,150),NZ
```

L'istruzione alla linea 80 contiene la PUT che richiama dalla matrice NZ il disegno ivi memorizzato, ricreandolo in una porzione equivalente (rettangolo di pari altezza e base) che viene localizzata con il suo solo vertice alto sinistro (figura 7.21). Si ricordi che l'origine degli assi e' in alto a sinistra. I dimensionamenti operati dalle relazioni alle linee 40 e 50 assicurano a sufficienza il corretto immagazzinamento degli attributi di una zona di 40x40 pixel. La GET, poi, con le coordinate dei vertici di un rettangolo, stabilisce quale effettiva porzione del video interessare. Dove immagazzinare gli attributi di questa porzione, sara' compito della matrice NZ, richiamata nella stessa GET.

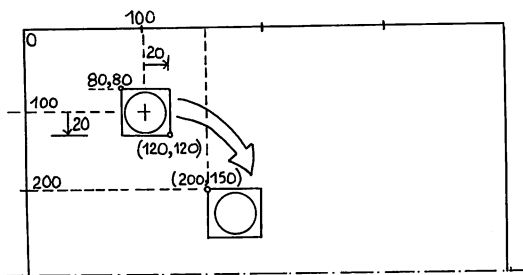



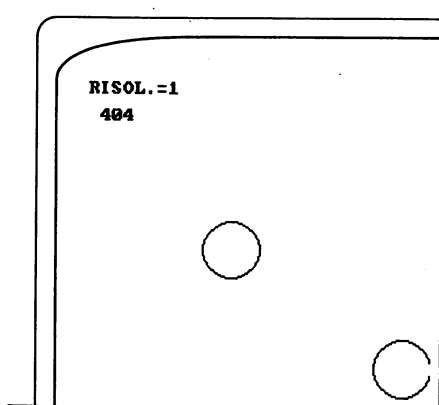
Figura 7.21

Una variante di dispo e' il seguente programma **dispo2**, in cui e' possibile scegliere la risoluzione e in cui si e' provveduto anche alla protezione dell'input per tale scelta. Il filtro consente di respingere tutti i valori diversi da 1, 2 e 3. Inoltre, poiche' solo per lo screen 1 e' necessaria una doppia densita' di scansione, in tal caso il filtro provvede anche ad assegnare a **p** il valore 2 (che verra' automaticamente assegnato nella linea 40). Alla linea 90, prima di riciclare a una nuova scelta di **z** e sperimentarne gli effetti, occorre azzerare la matrice NZ, altrimenti nascerebbe una situazione di errore causata dal fatto di voler dimensionare una matrice gia' esistente.

```
2 'dispo2:dispo a risoluz. var.
5 CLS:KEY OFF
7 INPUT "RISOL.=",Z$:Z=VAL(Z$)
8 IF Z<1 OR Z>3 THEN 7
9 IF Z=1 THEN P=2 ELSE P=1
10 SCREEN Z
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((P*H+7)/8)
50 DIM NZ(DZ)
60 GET (80,80)-(120,120),NZ
70 PUT (200,150),NZ
80 QQ$=INPUT$(1)
90 ERASE NZ:GOTO 7
```

Nel successivo programma **dispo31** e relativa hard-copy si puo' notare l'effetto di una GET insufficiente a scandire tutti i pixel della

circonferenza. Il motivo di cio' e' attribuibile all'ordinata del secondo vertice della GET che, anziche' a 120 e' stato piazzato a 119. Il numero che compare in alto e' il valore calcolato degli elementi che il vettore NZ e' capace di contenere.

```
2 'dispo31:dispo a risoluz. var.
5 CLS:KEY OFF
7 INPUT "RISOL.=",Z$:Z=VAL(Z$)
8 IF Z<1 OR Z>3 THEN 7
9 IF Z=1 THEN P=2 ELSE P=1
10 SCREEN Z
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((P*H+7)/8)
45 PRINT DZ
50 DIM NZ(DZ)
60 GET (80,80)-(119,120),NZ
70 PUT (200,150),NZ
80 QQ$=INPUT$(1)
90 ERASE NZ:GOTO 7
```

In **dispo4**, finalmente, troviamo i primi tentativi di **animazione**, tutti sperimentabili in qualsiasi risoluzione grafica. La riproduzione del disegno contenuto nel riquadro indicato nella GET e' a carico di una PUT il cui vertice di riferimento e' la variabile **x** di controllo del ciclo che incrementa di 5 pixel in ascissa ogni volta che viene sganciata la sospensiva alla linea 80. Infatti tale manovra riporta il programma alla linea 65, dove l'istruzione CLS pulisce lo schermo e cosi' via...

Per evitare di far funzionare la PUT in condizioni fuori schermo, per mezzo della linea 67, abbiamo protetto il programma limitando la x in funzione della risoluzione attiva.

```
2 'dispo4:dispo a risoluz. var.
5 CLS:KEY OFF
7 INPUT "RISOL.=",Z$:Z=VAL(Z$)
8 IF Z<1 OR Z>3 THEN 7
9 IF Z=1 THEN P=2 ELSE P=1
10 SCREEN Z
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((P*H+7)/8)
50 DIM NZ(DZ)
60 GET (80,80)-(120,120),NZ
65 CLS
67 IF X>Z*50 THEN X=-100
70 PUT (200+X,150),NZ
75 X=X+5
80 QQ$=INPUT$(1)
85 GOTO 65
90 ERASE NZ:GOTO 7
```

Nel programma dispo4 l'animazione e' stata ottenuta con la cancellazione dell'immagine e con la sua replica in una nuova posizione, mediante una PUT parametrica; in tal modo. pero', si ha la cancellazione di tutto lo schermo, mentre sarebbe sufficiente che fosse limitata alla sola porzione in cui lavora la PUT: per far cio' basta riscrivere un'altra PUT identica a quella che ha fatto riapparire l'immagine. In pratica, il programma **dispo4**, riarrangiato nella versione **dispo41**, crea lo stesso movimento senza cancellare tutto il video.

```
2 'dispo41:dispo a risoluz. var.
5 CLS:KEY OFF
7 INPUT "RISOL.=",Z$:Z=VAL(Z$)
8 IF Z<1 OR Z>3 THEN 7
9 IF Z=1 THEN P=2 ELSE P=1
10 SCREEN Z
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((P*H+7)/8)
50 DIM NZ(DZ)
60 GET (80,80)-(120,120),NZ
67 IF X>Z*50 THEN X=-100
70 PUT (200+X,150),NZ
71 QQ$=INPUT$(1)
72 PUT (200+X,150),NZ
75 X=X+5
85 GOTO 67
90 ERASE NZ:GOTO 7
```

La versione dispo411, infine, consente, attraverso alcune opzioni della PUT (PSET, PRESET, OR, AND, XOR) di sperimentarne gli effetti. In figura 7.21a si puo' vedere l'effetto dell'opzione **OR** aggiunta alla PUT della linea 72.

```
2 'dispo411:dispo a risoluz. var.
5 CLS:KEY OFF
7 INPUT "RISOL.=",Z$:Z=VAL(Z$)
8 IF Z<1 OR Z>3 THEN 7
9 IF Z=1 THEN P=2 ELSE P=1
10 SCREEN Z
20 CIRCLE (100,100),20
30 B=40:H=40
40 DZ=4+B*INT((P*H+7)/8)
50 DIM NZ(DZ)
60 GET (79,80)-(121,120),NZ
67 IF X>Z*50 THEN X=-100
70 PUT (200+X,150),NZ
71 QQ$=INPUT$(1)
72 PUT (200+X,150),NZ,OR
75 X=X+1
85 GOTO 67
90 ERASE NZ:GOTO 7
```

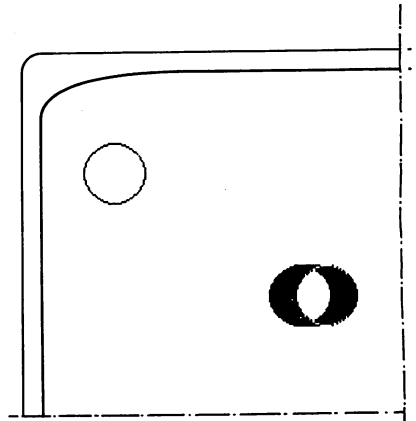



Figura 7.21a

7.4 SPIGOLATURE GEOMETRICHE

I programmi graduati che seguono hanno lo scopo di esaurire alcune interessanti opzioni dell'istruzione **circle**: si tratta di alcuni parametri con i quali, anziche' tracciare circonferenze, si possono disegnare **archi** e **settori** circolari. L'obiettivo grafico e' la determinazione di una procedura che tracci una scala graduata circolare in funzione di alcuni parametri esterni selezionabili dall'utilizzatore.

Ogni volta si puo' scegliere, ad esempio, il numero delle divisioni e l'angolo al centro. In effetti il risultato atteso e' solo un obiettivo intermedio. Puo' costituire infatti la base per una serie di esperimenti grafici per la simulazione di apparecchiature di misura a lettura analogica (ad indice). Il lettore puo' completare il programma inserendo una routine per la visualizzazione di un indice.

Iniziamo l'analisi del problema con un programma per la generazione di un poligono regolare, dato il numero dei lati; con l'occasione, richiamiamo un po' di trigonometria e, con riferimento alla figura 7.22, assumiamo il quadrato come base dei nostri ragionamenti.

Prendiamo come vertice base (0) un punto della circonferenza trigonometrica di ascissa pari al raggio; il prossimo vertice, preso nel senso antiorario, sara' ancora sulla circonferenza nel punto (1) di ascissa zero e ordinata pari al raggio. Il terzo vertice avra' coordinate (-R,0), e il quarto (0,-R).

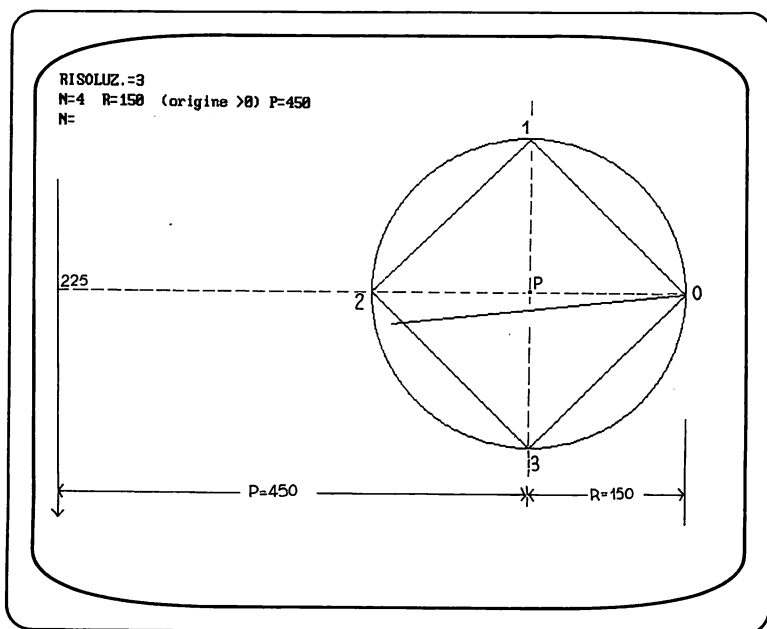



Figura 7.22

Per disegnare il quadrato in Basic ci serviamo di 4 istruzioni **line**, nella forma relativa, cioe' tutte aventi come punto iniziale l'ultimo punto tracciato.

Per la prima, pero', dovremo scegliere un punto di riferimento ad hoc con una **preset**, altrimenti verrebbe assunto implicitamente dalla macchina e sarebbe il centro del video.

Per analizzare con maggiore semplicita' il processo di generazione grafica non assumiamo nessun riferimento iniziale e quindi la situazione che si presentera' sara' quella di figura 7.22.

Segue il programma **poligo1** per la generazione di un qualsiasi poligono regolare.

```
10 'poligo1:generazione trigonometrica
20      '   con origine variabile.
21 CLS
22 KEY OFF
23 INPUT "RISOLUZ.=",Z
24 SCREEN Z
25 WINDOW (0,0)-(639,399)
30 PI#=3.1415926535#
40 INPUT; "N=", N
50 INPUT; "  R=", R
55 INPUT "  (origine >0) P=",P
60 A0=2*PI#/N
70 'PRESET (P+R,.5*P) 'origine nascosta!
80 FOR A=0 TO 2*PI# STEP A0
90 X=R*COS(A)
100 Y=.88*R*SIN(A)
110 LINE -(P+X,.5*P+Y)
120 NEXT A
130 A$=INPUT$(1)
140 GOTO 30
```

In testa al programma si propone la scelta della risoluzione. Alla linea 25 si riporta l'origine in basso a sinistra, in modo che il primo quadrante possa essere letto in modo tradizionale.

Si fissano, inoltre, le scale per far si' che, comunque scelga le coordinate della circonferenza trigonometrica, questa venga rappresentata nella stessa posizione relativa del video, indipendentemente dalla risoluzione assunta.

Inizialmente, proviamo a neutralizzare la linea 57, cosi' da mettere in evidenza che, senza l'appoggio della circle, si genera un segmento iniziale tra il centro del video e il punto **O**. Come parametri iniziali, si forniscono in ingresso il numero N di lati, il raggio R trigonometrico della circonferenza circoscritta e la posizione P del centro degli assi cartesiani rispetto ai quali si riferiscono le notazioni trigonometriche.

Si ricorda che per attivare la linea 57 basta sopprimere il carattere [']. Nel programma appare come costante il valore di PI (p greco), espresso in doppia precisione perche' conservi almeno le cifre indicate.

La linea scritta al passo 110 e' funzione della posizione di P e delle coordinate correnti X e Y, calcolate dalle seguenti funzioni trigonometriche:

X = R*cos(A)

Y = .83*R*sin(A)

dove:

.83 e' il coefficiente che tiene conto della diversa densita' verticale dei pixel

A e' la variabile di controllo del ciclo iterativo. Questo ciclo evolve **tante volte quanto e' il numero dei passi A0** in cui e' stato diviso l'angolo giro espresso in radianti.

Il valore di A0 risulta approssimato dalla divisione che compare nella linea 60. L'approssimazione, come vedremo, ha influenza sul risultato grafico di **chiusura** delle circonferenze.

Se calcoliamo, passo a passo, i valori di X e di Y nel caso N=4 R=55 e P=250 otteniamo i seguenti risultati:

A0 = 2*PI/4 = PI/2

per A = 0 (A e' il valore dell'angolo, espresso in radianti, tra il raggio vettore trigonometrico e l'asse delle ascisse)

X0 = R*cos(0) = 55*1

Y0 = .83*R*sin(0) = 0

0-1) per A = 2*PI/4 (equivale all'angolo retto)

X1 = 55*cos(2*PI/4) = 0

Y1 = .83*55*sin(2*PI/4) = .83*55 = 45.65

1-2) per A = PI/2 + PI/2 = PI (equivale all'angolo piatto)

X2 = -R = -55

Y2 = 0

2-3) per A = PI + PI/2 = 3*PI/2 (equivale all'angolo di 270 gradi)

X3 = 0

Y3 = .83*55*(-1) = -45.65

3-0) per A = 3*PI/2 + PI/2 = 2*PI (equivale all'angolo giro)

X4 = 55

Y4 = 0

Per evitare il segmento spurio iniziale B-0 e' sufficiente inserire nella linea 70 l'istruzione **preset(P+R,.5*P)** che regola la base di riferimento nel punto (0), oppure attivare il cerchio alla linea 57. In figura 7.23 troviamo alcune passate di poligo1.

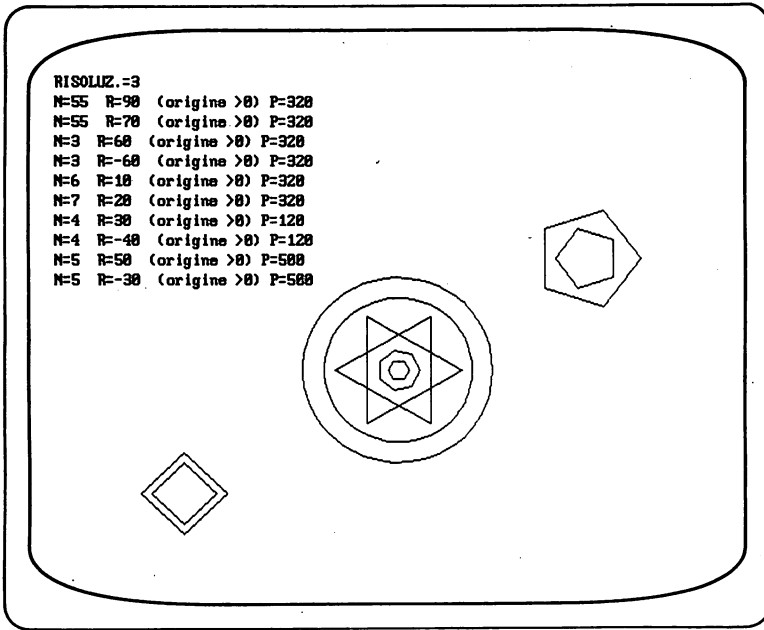



Figura 7.23

Si noti che i poligoni con numero di lati dispari ruotano invertendo il segno del raggio. In figura 7.24 riportiamo famiglie di poligoni concentrici, ottenuti ciclando su R, come appare dal seguente listato di **polimer**.

```
10 'polimer:generazione concentrica
20 CLS:KEY OFF
30 INPUT"RISOLUZ.=",Z:SCREEN Z
50 PI=3.1415926535#
60 INPUT"N= ",N
70 INPUT" R= ",R
80 INPUT" P= ",P
90 FOR B=1 TO 10
100 R=R+5
110 AO=2*PI/N
120 PRESET (P+R,.5*P)
130 FOR A=0 TO 2*PI STEP AO
140 X=R*COS(A)
150 Y=.87*R*SIN(A)
160 LINE -(P+X,.5*P+Y)
170 NEXT:NEXT:A$=INPUT$(1):GOTO 50
```

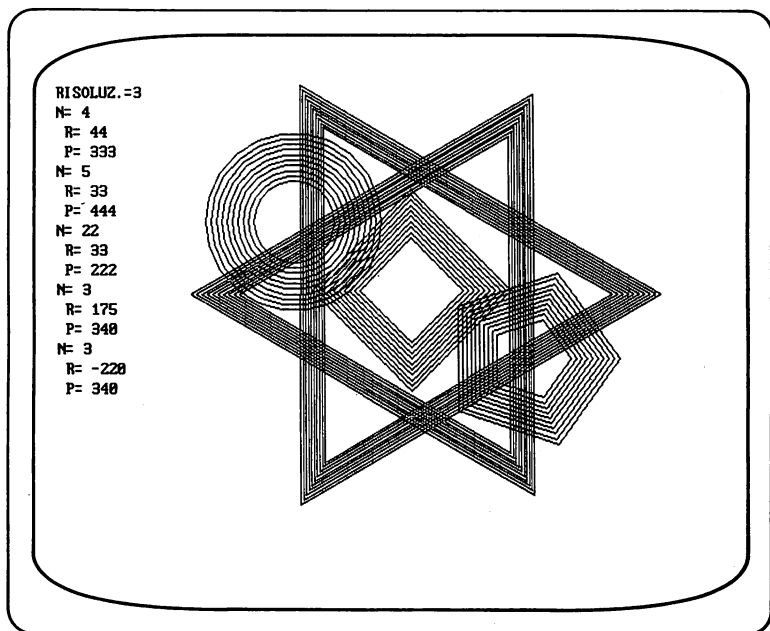



Figura 7.24

Trattiamo, infine, il programma **poligo20**, l'ultimo di una lunga serie di esperimenti per la generazione di archi e settori circolari: rispetto a poligo1, da cui discende, le novita' sono rappresentate dai nuovi parametri, attivati nella sintassi geometrica delle circle scritte alle linee 125 e 128. Queste circle, traccianti le circonferenze che delimitano la scala dello "strumento", contengono, dopo la dichiarazione del raggio, due variabili **I** e **F**. Tali parametri, indicanti rispettivamente inizio e fine dell'arco, servono appunto per tracciare parti di circonferenze o archi. I valori vanno introdotti con riferimento agli angoli, espressi in radianti. Si noti che introducendo i valori negativi di tali parametri la circle traccia anche i raggi congiungenti il centro con le estremita' dell'arco.

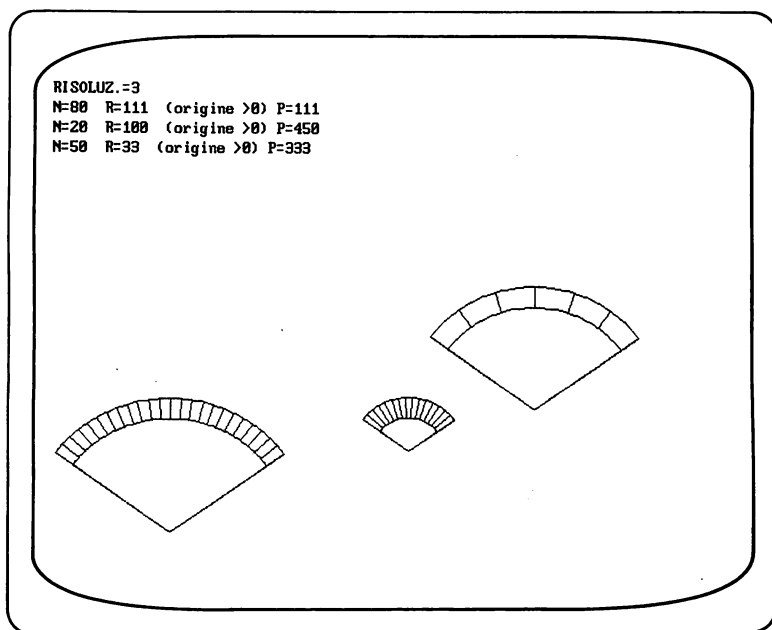
Esperimenti interattivi di generazione di archi possono essere condotti con il programma **arvar1**.

```
10 'arvar1:.....archi e settori variabili
20 CLS:KEY OFF:INPUT "Risol.=",Z:SCREEN Z
40 INPUT "iniz.arco --> ",I '-0.4
50 INPUT "fine arco --> ",F '-2.8
60 CIRCLE (300,150),180,,I,F
65 CIRCLE (300,150),170,,I,F
70 INPUT "",W:GOTO 20
```

Segue **poligo20** insieme ad alcune sue esecuzioni (figura 7.25). Si noti, alla linea 110, una singolare espressione di line utilizzata per il tracciamento delle "tacche" dello strumento: le coordinate delle estremita' del segmento sono variabili i cui valori, calcolati delle istruzioni precedenti, coincidono con quelli delle coordinate appartenenti alle circonferenze del quadrante.

```
10 'poligo20:generazione quadrante strumento
20 CLS:KEY OFF
23 INPUT"RISOLUZ.=",Z:SCREEN Z
25 WINDOW (0,0)-(639,399)
30 PI#=3.1415926535#
40 INPUT; "N=", N          '40
50 INPUT; "  R=", R        '200
55 INPUT "  (origine >0) P=",P'280
60 AO=2*PI#/N
70 PRESET (P+R,.3*P) 'origine nascosta!
80 FOR A=PI#/5 TO (4/5)*PI# STEP AO
95 X=R*COS(A):X1=(R+20)*COS(A)
105 Y=.83*R*SIN(A):Y1=.83*(R+20)*SIN(A)
110 LINE (P+X1,.3*P+Y1)-(P+X,.3*P+Y)
120 NEXT A
121 I=-36/57.295778#
122 F=-144/57.295778#
125 CIRCLE (P,.3*P),R+20,,I,F
128 CIRCLE (P,.3*P),R,,I,F
130 A$=INPUT$(1):GOTO 30
```

Figura 7.25

CAPITOLO
OTTO

IL SUONO E IL COLORE

Alla grafica, come mezzo espressivo di astrazioni logiche e matematiche , di cui abbiamo dato fin qui numerosi esempi elementari, tentiamo ora di accostare, con lo stesso metodo, anche i suoni e il colore.

Per fare esperimenti sonori dovremo accontentarci delle capacita' minime di M24, cioe' espressioni a una sola voce, mentre per esercitarsi con l'ambiente grafico a colori e' opportuno disporre di uno schermo colorato.

Obiettivo comune in questo laboratorio multimodale sara' quello di mettere i sensi in competizione tra loro per realizzare un prodotto elementare integrato, che possa servire sia a scopo educativo del linguaggio informatico, sia per acquisire sensibilita' logica sul piano artistico. Si dovrebbe arrivare, pertanto, a una riduzione simbolica comune, in cui l'esperienza sensoriale e l'intuizione artistica si fondono in un certo contesto matematico e si spingono verso espressioni razionali equivalenti.

8.1 PRIMI PASSI COL COLORE

Se si dispone di un video a colori le modalita' di usarlo cambiano a seconda di come abbiamo impostato alcune istruzioni BASIC che ne controllano le attivita'. Sappiamo gia', ad esempio, che l'istruzione SCREEN Z (con Z= 0, 1, 2, 3) seleziona la risoluzione grafica. Nella configurazione M24 cui facciamo riferimento, quando Z=0, si dice che il video lavora in modo "testo": cio' significa che non potremo usare istruzioni grafiche come line, circle, ... ma solo istruzioni per visualizzare caratteri del repertorio ASCII. Nel "modo testi" (Z=0) si puo' usare il colore per colorare i caratteri che devono apparire sul video: ciascuno dei 256 caratteri della tabella ASCII puo' quindi essere scritto a colori su fondo di colore diverso. Si noti che se il colore di fondo e' uguale a quello di scrittura il carattere non puo' apparire. Le possibilita' di colorare i caratteri in "modo testi" sono le combinazioni di 16 colori su una scelta di 8 colori di fondo. Inoltre, ciascuno dei 16 colori puo' essere usato in modo lampeggiante (blinking).

Per quanto riguarda la colorazione in ambiente grafico, non disponendo di una configurazione M24 dotata di particolari estensioni elettroniche per trattare tutti colori nella massima risoluzione, dovremo accontentarci di trattare solo una parte di questi. In particolare dovremo usare solo la risoluzione piu' bassa (Z=1), quella cioe' in cui il video puo' contenere in modo misto sia 25 righe da 40 caratteri, sia un'area grafica di 320 x 200 pixel. In questo ambiente grafico faremo le nostre esperienze colorate; il video verra' attivato nel programma con una **SCREEN 1** ed entrera' in attivita' un sistema di controllo degli attributi grafici, ivi compreso il colore.

Cosi', ad esempio, a ogni pixel sono associati 2 bit per il colore: cio' consente di selezionare (e memorizzare) fino a 4 colori da una tavolozza di otto. Si noti che tale organizzazione permane anche su una macchina con video monocromatico; in tal caso saranno visibili 4 toni di grigio. La stessa situazione risulta sulla stampante PR 15-B, quando si trasferisce in copia il contenuto del video con attributi colorati. Ovviamente a ogni tono di grigio corrisponde una propria densita' di punti battuti.

8.2 I TESTI COLORATI

In "modo testi" l'istruzione che consente la selezione del colore (o del tono di grigio) e':

```
COLOR s,f
```

dove **f** e' il numero (da 0 a 7) corrispondente al colore che si desidera avere come fondale e **s** quello del colore di scrittura. Questi ultimi con lo stesso numero di riferimento (da 0 a 7) selezionano gli otto colori di base, ma con i numeri da 8 a 15 ne selezionano un tono piu' brillante, mentre con i numeri da 16 a 31 si puo' ripetere la serie in modo lampeggiante. La tabella di figura 8.1 riporta i 16 colori con i rispettivi numeri di riferimento.

Numero	Colore	Numero	Colore
0	Nero	8	Grigio
1	Azzurro	9	Azzurro brillante
2	Verde bottiglia	10	Verde bandiera
3	Celeste	11	Celeste brillante
4	Rosso	12	Rosso brillante
5	Violetto	13	Violetto chiaro
6	Giallo oro	14	Giallo paglierino
7	Bianco	15	Bianco brillante

Figura 8.1

Cosi' il programma **szero**:

```
10 'szero
20 CLS:KEY OFF:S=0
30 INPUT F
50 SCREEN 0
55 PRINT
60 COLOR S,F
70 PRINT S;CHR$(1);
80 QQ$=INPUT $(1)
82 IF QQ$=" " THEN 20
85 S=S+1
90 GOTO 60
```

contiene un'istruzione **COLOR s,f**, in cui il fondale f viene scelto con la input di linea 30, mentre i colori di scrittura vengono introdotti con il valore del parametro s che a ogni ciclo viene incrementato di 1. Di conseguenza i due caratteri visualizzati dalla print alla linea 70 avranno colori corrispondenti. Ad esempio quando alla linea 60 (come da tabella in figura 8.1) si forma una **color 12,6**, la print visualizza caratteri in **rosso brillante su fondo giallo oro.**

Con il seguente programma **color2** si possono scegliere i seguenti parametri: l'entrata decimale ASCII del carattere che si desidera stampare; il colore del fondo su cui scrivere e la successione dei numeri di colore con cui si ripetera' la scrittura del carattere ASCII prescelto. Tale carattere, in virtu' della string$ di linea 220 verra' anche ripetuto 80 volte, con lo stesso colore di scrittura, sulla stessa riga.

```
10 'color2:scorrono gli ASCII
15 SCREEN 0,0,0:COLOR 14,0
20 CLS:KEY OFF:INPUT "E.DEC=",X
25 INPUT "fondo=",Y
30 INPUT A,B,C,D,E,F,G,H
40 COLOR A,Y:GOSUB 220
50 COLOR B,Y:GOSUB 220
60 COLOR C,Y:GOSUB 220
70 COLOR D,Y:GOSUB 220
80 COLOR E,Y:GOSUB 220
90 COLOR F,Y:GOSUB 220
100 COLOR G,Y:GOSUB 220
110 COLOR H,Y:GOSUB 220
120 COLOR H+8,Y:GOSUB 220
130 COLOR G+8,Y:GOSUB 220
140 COLOR F+8,Y:GOSUB 220
150 COLOR E+8,Y:GOSUB 220
160 COLOR D+8,Y:GOSUB 220
170 COLOR C+8,Y:GOSUB 220
180 COLOR B+8,Y:GOSUB 220
190 COLOR A+8,Y:GOSUB 220
200 W$=INPUT $(1)
210 GOTO 40
220 PRINT STRING$(80,X);
230 A$=INKEY$:IF A$=" " THEN 20
240 RETURN
```

Esaminiamo, infine, il seguente programma **degris** che permette di fare esperimenti anche in funzione della densita' di caratteri per riga.

```
10 'degris :PROVE DI COLORE SUL TESTO
15 CLS:KEY OFF
20 SCREEN 0,0,0:COLOR 15,0:LOCATE 1,1
25 CLS:PRINT "(a)=80 car/riga    (b)=40 car/riga"
26 Q$=INPUT $(1):IF Q$<>"a" AND Q$<>"b" THEN 26
27 IF Q$="a" THEN L=80 ELSE L=40
```

```
28 WIDTH L
30 COLOR 15,0
40 LOCATE 1,1:PRINT "Screen 0:Width"L"(car/riga)         ":COLOR 15,0
50 '          schermo alfanumerico(0): 80 car. per riga: bianco(7) su nero(0)
60 PRINT
70 LOCATE 2,1:PRINT"color Scrive 0...31, su Fondo 0...7 "
72 LOCATE 3,13:PRINT"          ":LOCATE 3,1
75 INPUT"      S----> ",S:IF S>31 THEN 72
80 DATA 7,6,3,2,5,4,1,0
82 FOR I=0 TO 7:READ F(I):NEXT
84 FOR I=0 TO 7
90 COLOR S,F(I)
100 PRINT "             ";S;"        FONDO ";F(I)
105 RESTORE :NEXT
115 QQ$=INPUT$(1):IF QQ$=" " THEN 20'.......batti spazio per tornare daccapo
120 IF (S=0 OR S=16) THEN 30 ELSE GOTO 72
```

Come si puo' vedere alla linea 28 compare l'istruzione WIDTH L, che significa larghezza di scrittura pari al valore del parametro L. In "modo testi" tale parametro puo' avere solo due valori 40 oppure 80, corrispondenti ad altrettanti caratteri da stampare su ciascuna delle 25 righe del video.

E' per questo che il programma contiene una selezione filtrata dalle linee 26 e 27 su quei due unici valori. Ovviamente, anche in **degris** si puo' scegliere il colore di scrittura del carattere (linea 75), mentre gli otto colori di fondo sono preregistrati in una lista F(i) di variabili con indice caricata dal ciclo alla linea 82. Il programma **degris** consente cosi' di fare esperimenti introducendo il valore del colore di scrittura con cui riprodurre un stesso testo che va a piazzarsi sul video in corrispondenza di 8 fondali diversamente colorati.

Le scelte possibili da programma riguardano, quindi, oltre ai 31 modi possibili di scrittura, 16 dei quali lampeggianti, anche la larghezza del carattere: 40 o 80 caratteri per riga.

Si notino nel programma le varie protezioni adottate, inclusa quella alla linea 120 che evita l'oscuramento totale del video quando si selezionano i numeri corrispondenti a scrittura in nero su fondo nero.

Al termine del ciclo, battendo la barra spaziatrice, si ritorna alla selezione della larghezza di scrittura, altrimenti, con qualsiasi altro tasto, si introducono altri valori per il colore di scrittura.

8.3 I GRAFICI COLORATI

Con la grafica a colori il video cambia natura e molte delle convenzioni introdotte per la modalita' "testi" devono essere rivedute; abbiamo allora costruito un piccolo programma-sonda, denominato **colz**, mediante il quale, insieme alle nuove definizioni possiamo sperimentare anche le precedenti gia' apprese e valutarne le differenze:

```
10 'colz
20 CLS:KEY OFF:S=0
30 INPUT "RISOL.=",Z
40 INPUT "PRESEL.",D
50 SCREEN Z
55 PRINT
60 COLOR S,D
70 PRINT S;CHR$(1);
80 QQ$=INPUT $(1)
82 IF QQ$=" " THEN 20
85 S=S+1
88 IF (Z=2 OR Z=3) AND S>3 THEN 20
90 GOTO 60
```

La prima differenza notevole e' quella che gli attributi del colore possono essere sfruttati solo a bassa risoluzione (SCREEN 1); la seconda differenza riguarda la sintassi dell'istruzione COLOR F,D alla linea 60. In **modo grafico** la color impiega due parametri che sono di significato completamente diverso da quelli della color in **modo testi**: qui F, con lo stesso significato di numero associato al colore di fondo, appare come primo parametro. I numeri dei colori di fondo da assegnare al parametro F si possono ricavare ancora dalla tabella di figura 8.1. Il parametro (D) ha ora lo scopo di preselezionare il primo o il secondo gruppo di tre colori con i quali si possono disegnare grafici o colorare interni di figure chiuse, come vedremo meglio in seguito. A parita' di preselezione D, la scelta del colore con cui disegnare viene fatta assegnando il valore 1, 2, 3 al parametro colore (k), all'interno della singola istruzione grafica.

Nella figura 8.2 riportiamo la tabella di riferimento per la scelta del parametro k in funzione della preselezione da effettuare nella **color** a monte. La manualistica corrente e' solita chiamare il parametro di preselezione "tavolozza" o "palette".

Scelta Colore (K=)	Preselezione (D=0)	Preselezione (D=1)
1	Verde	Celeste
2	Rosso	Violetto
3	Giallo	Bianco

Figura 8.2

Mettiamo in pratica l'esercizio delle convenzioni dettate dalle tabelle 8.1 e 8.2 col seguente programma **bcol8** di cui riportiamo la copia di alcune passate nella figura 8.3 e che consiste nel disegnare e colorare 5 rettangoli.

```
10 'bcol8:prova di colore
20 CLS:KEY OFF
30 SCREEN 1
35 LOCATE 1,15:Y=0:PRINT STRING$(20,32)
38 LOCATE 1,1
40 INPUT"fondo,presel.=",F,D
50 COLOR F,D 'd=presel.  su f= colore di fondo
60 INPUT "colorbox = ",K 'k=colore int. figura
70 LINE (50,100+12*Y)-(80,110+12*Y),K,BF
80 LINE (83,100+12*Y)-(113,110+12*Y),K+1,BF
90 LINE (116,100+12*Y)-(149,110+12*Y),K+2,BF
100 LINE (152,100+12*Y)-(185,110+12*Y),K+3,BF
110 LINE (188,100+12*Y)-(221,110+12*Y),K+4,BF
115 PRINT"premi spazio per altro fondo"
120 QQ$=INPUT $(1)
122 IF QQ$=" " THEN 35
130 Y=Y+1
140 GOTO 60
```

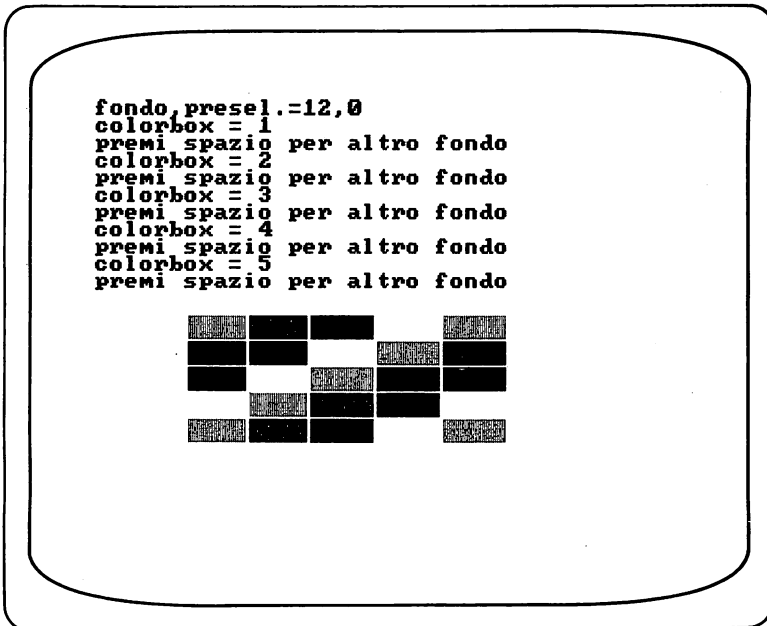



Figura 8.3

Qui i tre colori preselezionati con D=0 nella input di linea 40 e che vengono attualizzati con i valori assegnati al parametro k che compare nelle **line** vengono individuati con tre densita' di stampa per punti: la piu' debole corrisponde al verde, l'intermedia al rosso e la massima al giallo oro. Tutto viene disegnato su un fondo rosso brillante. La scrittura avviene in giallo oro. Vediamo l'effetto di una prima banalissima variante ai dati introdotti. Cambiamo il numero F del colore di fondo: anziche' assegnare il valore 12 corrispondente al rosso brillante (vedasi la tabella di figura 8.1), assegniamo il valore 4, corrispondente al rosso. Il risultato e' che non apparira' piu' il rettangolo colorato in rosso (come il fondo!). Se invece scegliamo D=1, coloreremo i rettangoli con l'altro gruppo di tre colori: celeste, violetto e bianco.

Altre esperimenti di intersezione logico-grafica possono essere condotti col seguente **bcol3**, dove la novita' consiste nell'aver messo in campo, oltre ai gruppi di rettangoli, una circonferenza in cui al parametro **K** viene assegnato il valore del colore di fondo (K=F); in tal modo si possono studiare le situazioni che nascono quando la circonferenza attraversa le zone diversamente colorate dei rettangoli.

Le scelte possibili avvengono, interattivamente, anche sui risultati gia' raggiunti: e' sufficiente battere la barra spaziatrice per tornare a scegliere il colore di fondo. Oppure, battendo il carattere [/], si puo' tornare daccapo e studiare altre strade di composizione grafica.

Nella figura 8.4 appare la situazione creatasi con le scelte ivi riportate. I segni meno marcati corrispondono al verde, quelli mediamente marcati al rosso e quelli piu' marcati al giallo.

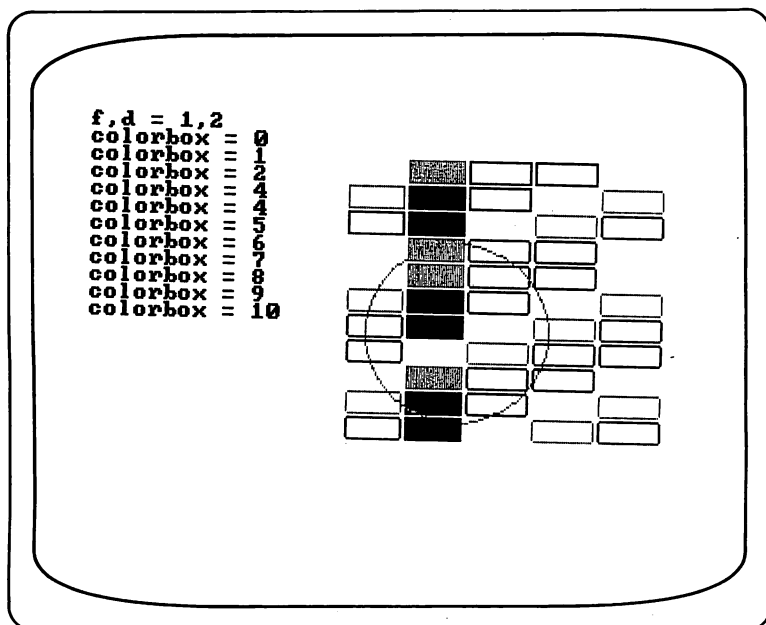



Figura 8.4

```
10 'bcol3:prova di colore
20 CLS:KEY OFF:SCREEN 0,0,0
30 SCREEN 1:Y=0
40 INPUT "f,d = ",F,D
50 COLOR F,D      'd=presel. su f=fondo
60 INPUT "colorbox = ",K
65 CIRCLE (200,100),50,F
70 LINE (140,20+12*Y)-(170,30+12*Y),K,B
80 LINE (173,20+12*Y)-(203,30+12*Y),K+1,BF
90 LINE (206,20+12*Y)-(239,30+12*Y),K+2,B
100 LINE (242,20+12*Y)-(275,30+12*Y),K+3,B
110 LINE (278,20+12*Y)-(311,30+12*Y),K+4,B
120 QQ$=INPUT $(1)
130 Y=Y+1
140 IF QQ$=" " THEN 20
145 IF QQ$="/" THEN 30
150 GOTO 60
```

Il successivo programma **bcol92** e' simile a bcol3, ma e' piu' sofisticato come manovre suggerite dalla macchina: si evitano, cosi', i passi incerti da compiere, di volta in volta. Un altro risultato prezioso e' che le scelte fatte appaiono subito sul video dentro le istruzioni che producono i risultati grafici (figura 8.5).

```
10 'bcol92'prova di colore
20 CLS:KEY OFF:SCREEN 0,0,0
30 SCREEN 1
35 LOCATE 3,1:PRINT STRING$(40,32)
36 LOCATE 6,1:PRINT STRING$(40,32)
37 LOCATE 7,1:PRINT STRING$(40,32)
40 LOCATE 1,1:F$="":Y=0:GOSUB 500
42 LOCATE 1,10:PRINT STRING$(20,32)
45 LOCATE 1,1:F$=""
50 INPUT"fondo   =",F$:F=VAL(F$)
60 IF F>31 THEN 35
65 GOSUB 500
66 LOCATE 2,10:PRINT STRING$(20,32)
70 LOCATE 2,1:INPUT"presel. =",D$:D=VAL(D$)
72 IF D$="" THEN 66
75 IF D>1 THEN 66
77 GOSUB 500
78 IF D$="n" THEN 40
80 COLOR F,D 'd=preselezione col.; f= colore di fondo
82 LOCATE 6,1:PRINT"premi <spazio> per colorare figure"
85 QQ$=INPUT $(1)
87 IF QQ$<>" " THEN 70
90 LOCATE 3,10:PRINT STRING$(30,32)
100 LOCATE 3,1:INPUT "colorbox=",K$:K=VAL(K$)'FIG.
101 LOCATE 3,14:PRINT STRING$(28,32)
102 IF K>31 THEN GOTO 90
105 CIRCLE(150,120),35,D
108 CIRCLE(150,120),45,F:PAINT (150,120),45,D
```

```
110 LOCATE 3,14:PRINT"line(x1,y1)-(x2,y2),"K",BF"
120 LINE (50,75+12*Y)-(80,85+12*Y),K,BF
130 LINE (83,75+12*Y)-(113,85+12*Y),K+1,BF
140 LINE (116,75+12*Y)-(149,85+12*Y),K+2,BF
150 LINE (152,75+12*Y)-(185,85+12*Y),K+3,BF
160 LINE (188,75+12*Y)-(221,85+12*Y),K+4,BF
170 LOCATE 6,1
180 PRINT"premi <spazio> per altro fondo          "
185 PRINT"PREMI (/) PER RICOMINCIARE"
190 QQ$=INPUT $(1)
195 Y=Y+1
200 IF QQ$=" " THEN 35
210 IF QQ$="/" THEN 20
220 GOTO 90
500 LOCATE 1,14:PRINT STRING$(25,32)
505 LOCATE 1,14:PRINT"color ("F","D")"
510 RETURN
```

Le linee da 120 a 160 disegnano 5 rettangoli (BOX colorati in una sequenza dipendente da k, che viene scelto dalla input alla linea 100). Ad ogni ciclo delle istruzioni comprese tra le linee 90 e 220 si puo' cambiare K e siccome ogni volta il contatore in 195 incrementa di 1 la Y che influisce sulla posizione dei rettangoli generati, il risultato precedente puo' essere confrontato col nuovo.

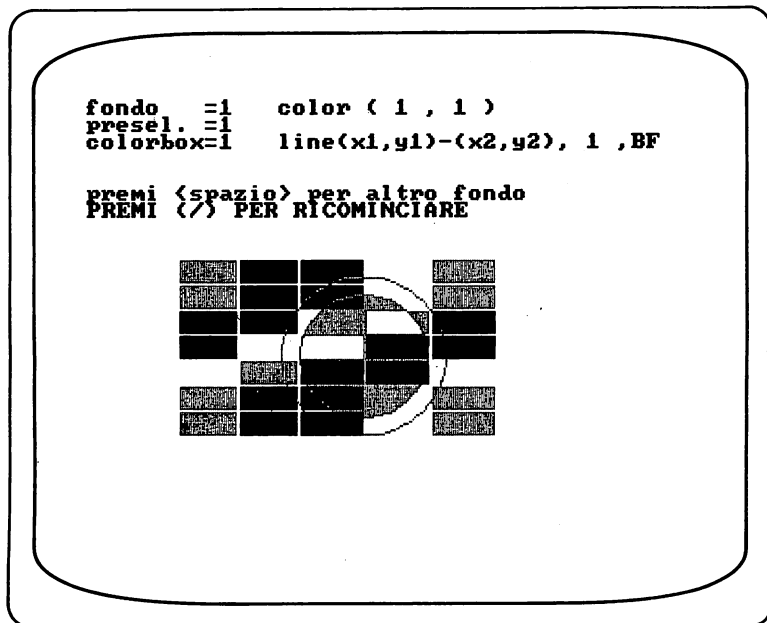



Figura 8.5

Infine, il successivo programma **merav1** integra le nozioni sui colori di fondo e le intersezioni tra figure colorate in ambiente finestre.

Gli effetti prodotti da alcune passate sono visibili in figura 8.6: con i valori introdotti gli unici colori presenti sono il celeste utilizzato per colorare i caratteri, il fondo della finestra, la circonferenza inferiore e l'ellisse, e l'azzurro per il fondale principale e l'area intorno all'ellisse (tutto cio' che appare bianco sulla pagina).

Provate ora a battere il tasto [/] e vedrete come cambia lo scenario: i colori si alternano. A un certo momento l'ellisse, la circonferenza maggiore e la cornice della finestra sono celesti, mentre quella minore e il fondale principale sono azzurri e il fondo della finestra e' violetto.

Se avessimo introdotto il fondo 12 con preselezione 0, il colore 1 per la circonferenza e lo zero per la paint, il risultato sarebbe stato il rosso brillante (12) per il fondale del video e l'area compresa tra la circonferenza maggiore e l'ellisse, il verde per la circonferenza minore e l'area dell'ellisse, mentre i caratteri in giallo oro su campo rosso.

A volte i risultati sono l'intersezione logica dei colori assegnati ai pixel tra loro piu' vicini, come nella figura 8.7, dove l'area colorata di celeste non e' quella dell'ellisse, ma l'area comune dell'ellisse e della circonferenza minore. Con **merav1**, quindi, si possono studiare delle condizioni logiche verificabili come dei veri diagrammi di Venn.

```
10 'merav1:finestre colorate
20 SCREEN 0,0,0
30 CLS:KEY OFF:SCREEN 1:WIDTH 13
40 INPUT "fondo  ->",F '5=VIOL
50 INPUT "presel.->",S '4=GIAL
60 COLOR F,S
70 LOCATE 3,6:PRINT STRING$(4,32)
80 LOCATE 3,1:INPUT "circ=",C '    3=GIAL
90 LOCATE 4,6:PRINT STRING$(3,32)
100 LOCATE 4,1:INPUT "pitt=",P '    2=ROSS
110 WINDOW (0,0)-(639,399)
120 VIEW (120,5)-(300,150),,1
130 LOCATE 6,3:PRINT STRING$(4,32)
140 LOCATE 6,1:PRINT "P=";P
150 PAINT (50,50),P
160 CIRCLE (319,199),150,P
170 CIRCLE (450,199),120,1,,,1.9
180 PAINT (450,199),P
190 CIRCLE (319,199),300,C
200 A$=INPUT $(1):IF A$=" " THEN 20
210 IF A$="/" THEN P=P+1:GOTO 120
220 IF A$="f" THEN 50
230 GOTO 70
```

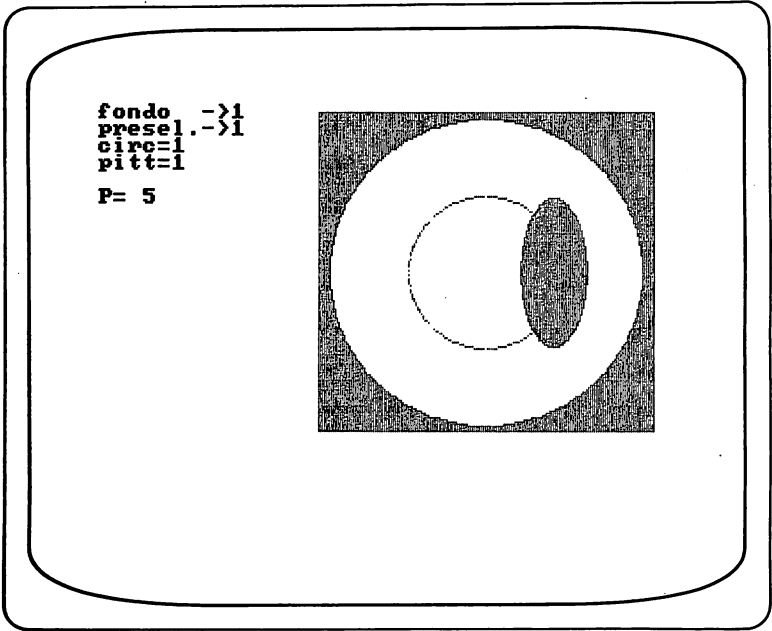



Figura 8.6

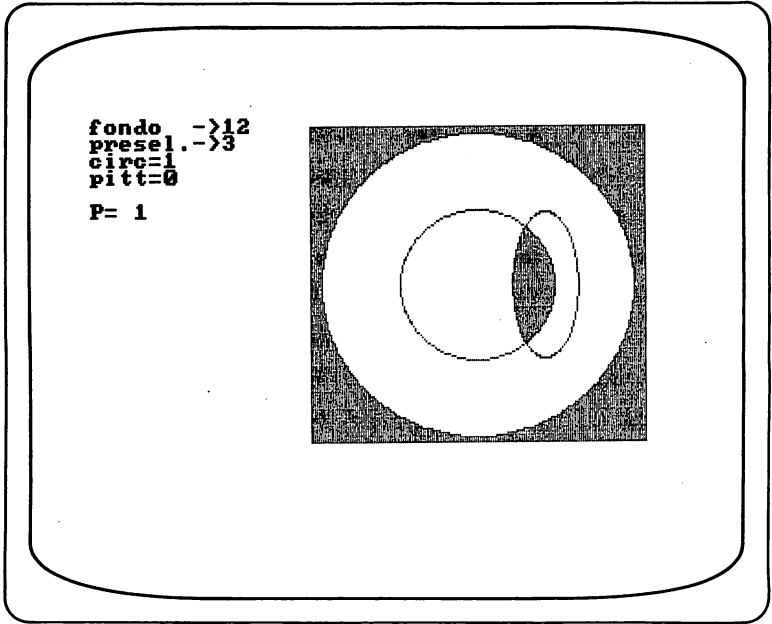



Figura 8.7

8.4 IL SUONO IN M24

Le istruzioni Basic mediante le quali il piccolo altoparlante di M24 produce suoni sono BEEP, SOUND e PLAY.

La prima, che e' la piu' semplice ed equivale a PRINT CHR$(7), produce un suono breve di altezza costante, mentre le altre due, SOUND e PLAY, possono generare suoni continui e a diverse altezze. La differenza tra queste due istruzioni, entrambe parametriche, e' che con **SOUND** si possono generare suoni di altezza (frequenza) variabile da 37 a 32767 Hz e di durata compresa tra circa 1/20 di secondo e 1/2 ora, mentre con **PLAY** si possono generare suoni di ben determinata altezza e durata corrispondenti a note o gruppi di note musicali, condensate, secondo opportune regole, negli elementi di una **stringa.**

Nei successivi paragrafi esamineremo le proprieta' fondamentali di queste istruzioni attraverso l'esercizio guidato di alcuni programmi parametrici che costituiscono un banco di prova per la costruzione di una specie di mini-linguaggio musicale.

8.5 LE FRASI PER IL CONTROLLO DEL SUONO

Iniziamo lo studio sistematico dell'istruzione SOUND con un piccolo programma (**otos1**) per l'esercizio di SOUND, che rendiamo variabile nei suoi due parametri: H (altezza) e D (durata). Tali grandezze vengono introdotte da opportune input alle linee 30 e 40.

```
10 'otos1:          prove di sound
20 KEY OFF:CLS:CLEAR
30 INPUT "altezza=",H '(max=32767)
40 INPUT "durata=",D
50 SOUND H,D
60 GOTO 30
```

In tal modo si possono controllare con un diapason e un cronometro.

Nel successivo programma **rotos** si puo' sperimentare un suono di date altezza (H) e durata (D) che si ripete 10 volte con una pausa tra un suono e l'altro. La pausa e' realizzata con un ciclo di ritardo all'istruzione 90, nell'ambito di un ciclo che realizza appunto il treno delle 10 note.

```
10 'rotos:        prove di sound con ripetizione
20 KEY OFF:CLS:CLEAR '(44,10,3000);(200,10,3000)
30 INPUT "altezza=",H '(min. 37; max=32767):300
40 IF H<37 THEN 30
50 INPUT "durata=",D ':3
60 IF D>10 THEN 50
70 INPUT "pausa=",P  'pausa tra i 10 suoni:3000
80 FOR R=1 TO 10
90 FOR I=1 TO P:NEXT
100 SOUND H,D
110 NEXT:GOTO 30
```

Con **rotos** e' importante acquisire sensibilita' agli ordini di grandezza delle variabili in gioco. Per ottenere effetti da "segnale orario" bisogna introdurre terne particolari di valori per H, D, P, ad esempio: 3000, 0.1, 3000. Il risultato e' rappresentabile attraverso la figura 8.8.

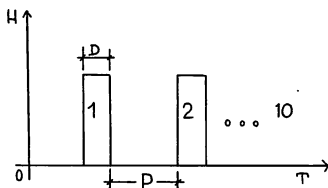



Figura 8.8

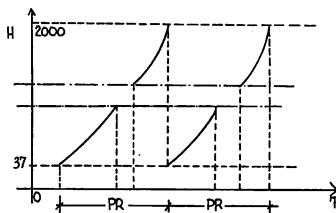



Figura 8.9

Derivato da **rotos** il successivo programma **siref** consente di creare un effetto sirena in funzione di un suono crescente (figura 8.9) in altezza con ragione **P** Hz. La serie delle note di durata costante .002 parte dalla frequenza 37 Hz e termina alla frequenza PR introdotta con la input alla linea 30. Valori accettabili per Pr e P sono, rispettivamente, 2000 e 5.

```
10 'siref:......... effetto sirena
20 KEY OFF:CLS:CLEAR  '  (2000,5)
30 INPUT "periodo=",PR      '2000
40 IF PR<37 THEN 30
50 INPUT "passo=",P      '(1<P<500)
60 FOR L=37 TO PR STEP P
70 SOUND L,.002:NEXT L:QQ$=INKEY$
80 IF QQ$=" " THEN 30 ELSE GOTO 60
```

Anche il successivo programma **(otos2)** deriva da **rotos** e produce effetti sonori con la possibilita' ulteriore di variare il passo di generazione (linea 60), a parita' di altezza massima e durata del singolo suono. Una terna accettabile per H, D, P sono i valori 2000, 1, 2000.

```
10 'otos2:  prove di effetti sonori
20 KEY OFF:CLS:CLEAR    '(222,2,22)
30 INPUT "altezza=",H    'max=32767
40 INPUT "durata=",D
50 INPUT "passo=",P
60 FOR L=37 TO H STEP P
70 SOUND L,D
80 QQ$=INKEY$
90 IF QQ$=" " GOTO 50
95 IF QQ$="i" THEN 30
100 NEXT L
110 GOTO 60
```

Infine, **otos** consente di fare un esame della sensibilita' dell'udito alla soglia superiore. Il suono emesso dal SOUND alla linea 70 ha una

durata brevissima, ma un'altezza crescente in funzione del passo P del ciclo di generazione stabilito dalla linea 60. Il codice generato dalla pressione del tasto spaziatore, rivelato dal confronto alla linea 90, consente di saltare fuori dal ciclo e stampare il valore dell'altezza del suono al momento del salto: se quello era il primo momento in cui non si udiva piu' alcun suono il valore L stampato corrisponde alla soglia superiore di udibilita' dell'operatore. Il programma, inizialmente, richiede il valore massimo della frequenza del suono da generare e il passo, cioe' l'intervallo di frequenza tra due suoni contigui. Come valore massimo di altezza e' inutile introdurre valori superiori a 18000, soglia a cui un essere umano, normalmente, non riesce piu' a percepire alcun rumore, mentre per il passo dipende dalla precisione con cui il soggetto vuol misurare la **sua** soglia. Un valore medio per il passo e' 5, ma puo' essere interessante sperimentare anche un passo piu' piccolo quando si sospetti di avere un "buco" di insensibilita' prima della soglia massima. Con maggiore velocita' di salita, tale circostanza avrebbe potuto passare inosservata.

Il programma consente di ripetere la prova semplicemente battendo la barra spaziatrice; se, invece, durante l'esecuzione, la si vuole interrompere per tornare daccapo o solo per cambiare il passo, basta premere i tasti [i]=inizio o [p]=passo, rispettivamente.

```
10 'otos:......prove di audiometria
20 CLS:KEY OFF:CLEAR
30 PRINT
40 INPUT "altezza=",H    'max=32767
50 INPUT "passo=",P
60 FOR L=37 TO H STEP P
70 SOUND L,.002:SG$=INKEY$
80 IF SG$=" " THEN 110
90 IF SG$="p" THEN PRINT: GOTO 50
95 IF SG$="i" THEN 30
100 NEXT L
110 PRINT L;" ";
120 GOTO 60
```

Procedendo sistematicamente nello studio dei suoni esaminiamo ora il successivo programma, denominato **scatson** perche' genera suoni dando l'impressione di una progressione inscatolata di frequenze. Con il parametro L si sceglie l'estremo superiore di una serie di suoni che cresce in altezza con ragione 1 ad ogni ciclo, secondo il ciclo scritto alla linea 60.

Tale serie evolve a partire dal valore 246.94-L e poiche' la frequenza di lavoro di SOUND non puo' essere inferiore a 37 Hz (altrimenti insorge un errore) L al massimo potra' essere 246.94-37=210.

Scelte di L che producono suoni troppo lunghi possono essere interrotti premendo il tasto spaziatore. L'istruzione sound alla linea 60, nel corpo del ciclo, oltre ad avere l'altezza del suono che a ogni giro cresce di 1 Hz, ha anche la durata che diventa sempre piu' piccola. La pausa introdotta alla linea 70 (frequenza zero e' il silenzio) serve a staccare i suoni l'uno dall'altro: lo stacco, a sua volta, si riduce a ogni ciclo in funzione del valore attuale di k.

```
10 'scatson:    inscatolamento
20 CLS:KEY OFF' ........sonoro
30 CLEAR:INPUT "L=",L 'max=210
40 'IF L>210 THEN 30
50 FOR K=L TO 1 STEP -1
60 SOUND 246.94-K,K/20
70 SOUND 0,K/15
80 I$=INKEY$:IF I$=" " THEN 30
90 NEXT K
100 SOUND 800,8
110 SOUND 350,8
130 FOR K=2000 TO 550 STEP -10
140 SOUND K,K/4000 'caduta lib.
150 NEXT K
160 GOTO 30
```

Una rappresentazione non in scala della successione dei suoni con gli opportuni riferimenti ai numeri di linea interessati alla generazione dei vari suoni e' fornita dalla figura 8.10.

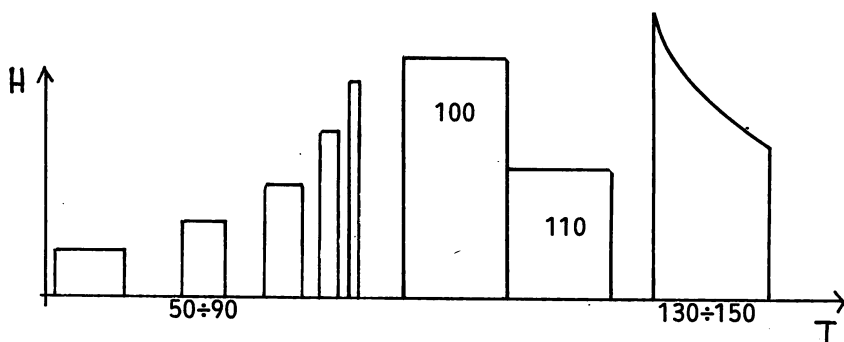



Figura 8.10

Nel successivo programma **scatson1** vediamo una versione di scatson con la modifica per la stampa dei valori delle frequenze e della durata dei suoni.

```
10 'scatson1:          inscatolamento
20 CLS:KEY OFF'        ........sonoro
30 CLEAR:INPUT "L=",L  '      max=210
40 IF L>210 THEN 30
50 FOR K=L TO 1 STEP -1
55 F=246.94-K:D=K/25
57 PRINT F;" ";D" ";
60 SOUND F,D
70 SOUND 0,K/15
80 I$=INKEY$:IF I$=" " THEN 155
90 NEXT K
```

```
100 SOUND 800,8
110 SOUND 350,8
130 FOR K=2000 TO 550 STEP -10
140 SOUND K,K/4000
150 NEXT K
155 PRINT
160 GOTO 30
RUN
L=20
 226.94   .8   227.94   .76   228.94   .72   229.94   .68   230.94   .64
 231.94   .6   232.94   .56   233.94   .52   234.94   .48   235.94   .44
 236.94   .4   237.94   .36   238.94   .32   239.94   .28   240.94   .24
 241.94   .2   242.94   .16   243.94   .12   244.94   .08   245.94   .04
```

Una variante di scatson1 e' rappresentata dal programma **scatson2** in cui la progressione dei suoni, sempre con ragione -1, questa volta decresce (nell'istruzione alla linea 55 ora c'e' un segno positivo).

```
10 'scatson2:           inscatolamento
20 CLS:KEY OFF'         ........sonoro
30 CLEAR:INPUT "L=",L  '      max=100
40 IF L>100 THEN 30
50 FOR K=L TO 1 STEP -1
55 F=246.94+K:D=K/25' 'F diminuisce
57 PRINT F;" ";D" ";
60 SOUND F,D
70 SOUND 0,K/15
80 I$=INKEY$:IF I$=" " THEN 155
90 NEXT K
100 SOUND 800,8
110 SOUND 350,8
130 FOR K=2000 TO 550 STEP -10
140 SOUND K,K/4000 'caduta lib.
150 NEXT K
155 PRINT
160 GOTO 30
RUN
L=20
 266.94   .8   265.94   .76   264.94   .72   263.94   .68   262.94   .64
 261.94   .6   260.94   .56   259.94   .52   258.94   .48   257.94   .44
 256.94   .4   255.94   .36   254.94   .32   253.94   .28   252.94   .24
 251.94   .2   250.94   .16   249.94   .12   248.94   .08   247.94   .04
```

Il programma **trill** consente di generare una successione di suoni di altezza crescente con ragione H scelta a piacere in modo da sperimentare le distanze a orecchio e verificare che, quando le distanze sono costanti, la successione non e' quella naturale dei toni musicali. La nota piu' bassa (f-261.83) e' il do in quarta ottava, quella piu' alta il do in sesta ottava. L'istruzione SOUND e' posta alla linea 70, insieme a una pausa (linea 80) per staccare i

suoni l'uno dall'altro, nel corpo di un ciclo che si ripete tre volte, ma ogni volta a partire dal valore di frequenza piu' alto raggiunto dal ciclo precedente. In tal modo il numero delle note a ogni ciclo si riduce, avendo fissato il limite superiore. Inoltre, poiche' i suoni generati oltrepassano l'estensione di un'ottava, si e' posto una condizione alla linea 50, che quando il valore della frequenza e' superiore a quella del do di quinta (524 Hz), la distanza H sia doppia. Si noti ancora che per valori di H superiori a 1046.5-261.83 (circa 784) il programma si riduce a emettere tre suoni di uguale altezza pari a 261.83. Infatti i cicli non scattano perche' il passo e' uguale all'intervallo. L'istruzione alla linea 90 visualizza il valore attuale delle frequenze del suono emesso.

```
10 'trill:for next per sound
20 CLS:KEY OFF
30 INPUT;"H= ",H:PRINT "  ";
40 FOR I=1 TO 3
50 IF J>524 THEN H=2*H
60 FOR J=261.83 TO 1046.5 STEP H
70 SOUND J,5
80 SOUND 0,5
90 PRINT " j("I")=";J;:NEXT:NEXT
95 PRINT:GOTO 30
```

Il successivo programma **temp** consente di costruire a orecchio la scala naturale dei toni musicali a partire dal do di quarta ottava (F=261.626). A tal fine bisogna introdurre la distanza giusta tra un tono e l'altro. La successione dei toni viene registrata nel vettore F(j) alla linea 140, mentre l'altezza del suono generato alla 170 viene stampato subito dopo alla linea 180. In qualunque momento si voglia rivedere la successione e verificare le distanze tra i suoni generati, basta battere la lettera [r]; battendo la lettera [i] si ripete tutto daccapo.

```
10 'temp:scala temperata ?
20 CLS :KEY OFF
30 PRINT:CLEAR
40 DIM F(200)
50 SOUND 261.626,8
60 F=261.626
70 INPUT;"h=",H$
80 IF H$="r" THEN 200
90 IF H$="i" THEN 30
100 H=VAL(H$)
110 F=F+H
120 F(0)=261.626
130 J=J+1
140 F(J)=F
150 JS=J
160 IF F>15000 THEN 30
170 SOUND F,8
180 PRINT "->F=";F;" ";
190 GOTO 70
```

```
200 FOR J=0 TO JS
210 SOUND F(J),3
220 SOUND 0,.8
230 NEXT:PRINT
240 FOR J=0 TO JS-1
250 D=F(J+1)-F(J)
260 PRINT D" ";
270 NEXT :QQ$=INPUT$(1)
280 IF QQ$="z" THEN 30
290 IF QQ$="i" THEN 200 ELSE GOTO 70
RUN
h=12->F= 273.626  h=13->F= 286.626  h=23->F= 309.626  h=45->F= 354.626  h=r
 12   13   23   45
```

Col successivo programma **temp1** possiamo verificare il nostro orecchio musicale messo alla prova dal precedente **temp.** Prima di commentare il programma e' opportuno pero' richiamare alcuni concetti di base del nostro sistema musicale (sistema temperato). Tale sistema e' basato sulla divisione artificiale dell'intervallo dell'ottava (intervallo tra suoni di altezza doppia) in dodici intervalli uguali detti semitoni; cio' da' luogo a una serie di dodici suoni diversi equidistanti, cioe' a una scala dodecafonica, detta anche piu' comunemente cromatica. I suoni della scala cromatica formano una progressione geometrica. L'intervallo costante che ne e' la ragione, cioe' il semitono, e' misurato dalla radice dodicesima di 2 (=circa 1.05946), per cui la serie numerica e' la seguente:

$$\mathbf{1},\ 2^{(2/12)},\ 2^{(3/12)},\ 2^{(4/12)},\ 2^{(5/12)},\ 2^{(6/12)},\ 2^{(7/12)},\ 2^{(8/12)}$$

$$2^{(9/12)},\ 2^{(10/12)},\ 2^{(11/12)},\ \mathbf{2.}$$

Abbiamo cosi' inserito 12 medi geometrici tra 1 e 2. Con questi moltiplicatori siamo in grado di ottenere qualsiasi altra nota del nostro sistema musicale.

```
10 'temp1:scala temperata ?
20 CLS :KEY OFF
30 PRINT:CLEAR
40 DIM F(200)
50 SOUND 261.626,8
60 F=261.626
70 PRINT
80 INPUT;"h=",H$
90 IF H$="r" THEN 220
100 IF H$="i" THEN 30
110 H=VAL(H$)
120 T#=2^(1/12)
130 F=F*T#
140 F(0)=261.626
150 J=J+1
160 F(J)=F
170 JS=J
```

```
180 IF F>15000 THEN 30
190 SOUND F,8
200 PRINT "->F=";F;" ";
210 GOTO 80
220 FOR J=0 TO JS
230 SOUND F(J),3
240 SOUND 0,.8
250 NEXT:PRINT
260 FOR J=0 TO JS-1
270 D=F(J+1)-F(J)
280 PRINT D" ";
290 NEXT
300 QQ$=INPUT$(1)
310 IF QQ$="i" THEN 30
320 IF QQ$="r" THEN 220 ELSE GOTO 70
RUN
h=->F= 277.1831  h=->F= 293.6652  h=->F= 311.1275  h=->F= 329.628  h=->F=
 349.2287  h=->F= 369.9949  h=->F= 391.9959  h=->F= 415.3052  h=->F= 440.0005
=->F= 466.1643  h=->F= 493.8838  h=->F= 523.2516  h=->F= 554.3658  h=->F=
 587.33  h=->F= 622.2544  h=->F= 659.2555  h=->F= 698.4569  h=->F= 739.9892  h
->F= 783.9912  h=->F= 830.6097  h=->F= 880.0002  h=_
```

In **temp2**, di cui riportiamo solo le prime linee che lo differenziano da temp1, la prima frequenza generabile e' quella piu' bassa consentita da SOUND che in scala cromatica conduce a un do centrale pari a 261.626 vibrazioni al secondo: tale frequenza e' quella riportata alla linea 50 e corrisponde alla nota musicale [RE -2].

```
10 'temp2:scala temperata ?
20 CLS :KEY OFF
30 PRINT:CLEAR
40 DIM F(200)
50 SOUND 38.89105,8
60 F=38.89105
```

Per chi abbia un po' di conoscenze di acustica si fa notare che la dissonanza nasce dai battimenti che si generano quando due o piu' note vengono suonate insieme formando un accordo: in armonia, la quinta in accordo naturale con un **la** di terza ottava (880 vibrazioni al secondo) e' un **mi** che deve vibrare 880 x 3/2= 1320 volte al secondo. Con la convenzione della scala cromatica il mi4 e' dato da:

$$880 \cdot 2^{7/12} = 880 \times 1.49831 = 1318.513.$$

Rispetto al mi4 naturale, la differenza (1320.000-1318.513=1.487), e', quindi, di circa 1.5 ogni secondo; cosi', suonando un la3 insieme a un mi4 temperato, si avranno 3 battimenti ogni due secondi. Questo e' il prezzo dell'aver posto un po' di ordine matematico tra le note musicali!

8.6 STRINGHE MUSICALI

L'istruzione PLAY e' pilotata da una sequenza di parametri che riguardano gli attributi delle note musicali rappresentate sinteticamente in una stringa, cioe', sostanzialmente, la loro altezza e durata. Siccome la stringa associata a un'istruzione PLAY e che ne costituisce l'argomento puo' riguardare piu' note, si intuisce la necessita' di costruire un tessuto ritmico e temporale al quale far riferimento.

Con l'istruzione PLAY si possono suonare 84 note, distribuite in 7 ottave. Per indicarle si faccia riferimento al seguente programma **equison.**

```
10 'equison:equivalenza
20 PLAY "n25"
30 SOUND 261.626,2
40 PLAY "o2"
50 PLAY "c"
60 PLAY "o2c"
70 PLAY "n37"
80 SOUND 523.251,2
90 PLAY "o3c"
```

La stringa **n25** nella PLAY di linea 20 significa il venticinquesimo semitono nella scala cromatica, avendo assunto il primo uguale al do in ottava zero (do-2). Lo stesso suono si puo' ottenere con l'istruzione sound per la frequenza 261.626 che e' la frequenza del **do** centrale ottenura rispettando la nota progressione geometrica.

Altro modo di usare la PLAY e' di dichiarare tra virgolette il nome della nota (in questo caso secondo la convenzione anglosassone c=do) e, poiche' si tratta di un "do" in seconda ottava, dichiarare anche questo attributo. Il tutto puo' essere racchiuso nella stessa stringa assegnata alla PLAY di linea 60. Analogamente "02c: il do in terza ottava, pari a 523.251 vibrazioni al secondo si puo' ottenere con le tre istruzioni 70, 80 e 90. Si noti che per ottenere l'ottava superiore della nota n25 basta aggiungere 12 (semitoni).

Facendo girare equison sentiamo, tuttavia, cinque note anziche 7: cio' dipende dal fatto che quelle contigue e uguali risultano legate e quindi indistinguibili. Per separarle modifichiamo equison in **equison1.**

```
10 'equison1:equivalenza
15 KEY 9,"cont"+CHR$(13)
20 PLAY "n25":STOP
30 SOUND 261.626,2
32 STOP
40 PLAY "o2"
50 PLAY "c":STOP
60 PLAY "o2c":STOP
70 PLAY "n37":STOP
80 SOUND 523.251,2:STOP
90 PLAY "o3c"
```

In equison1 si impiega anche il tasto funzionale F5 che genera la parola CONT, mediante la quale e' possibile far riprendere un programma interrotto da una istruzione STOP. Di queste istruzioni risulta riempito equison che le ha inserite immediatamente dopo ogni istruzione sonora: in tal modo con interruzioni e riprese forzate possono ora contarsi le sette note generate da equison.

Fin qui abbiamo acquisito le due modalita' di scrittura delle note nell'istruzione PLAY: quella numerale da 1 a 84 e quella musicale che connota le varie note secondo il metodo anglosassone che associa la prima lettera dell'alfabeto (a=la) alla nota **la** e continua nell'ordine alfabetico fini al **sol** (a, b, c, d, e, f, **g**). I diesis si ottengono facendo seguire alla nota il simbolo [ ], i bemolle dal simbolo [-]. IL mi=e e il si=b **non** possono essere dichiarati come **f-** e **c-**. Quando, nella notazione musicale, manchi l'indicazione dell'ottava, s'intende implicitamente che si desidera riferirsi all'ottava centrale [o3]. Il programma **sonvar1** impiega interattivamente l'istruzione PLAY: infatti, basta sostituire alla stringa musicale una o piu' variabili stringa, ciascuna preceduta da una **x.** Tali variabili vengono introdotte con opportune istruzioni di input. Gli esempi riportati nel commento dele input alle linee 20 e 30 sono le successioni di una scala in do maggiore [+] e una in do minore [-].

```
10 'sonvar1 :    stringhe di note
20 INPUT A$ 'o4cego5c:do+(maggiore)
30 INPUT B$ 'o4ce-go5c: do-(minore)
40 PLAY "xb$;xa$;"
45 W$=INPUT$(1)
47 IF W$=" " THEN 40 ELSE GOTO 20
```

In tal modo e' possibile ripetere un certo motivo annidato in una stringa, come avviene in sonvar1 quando, battendo la barra spaziatrice, si ritorna alla linea 40 dove nella PLAY non ci sono le due stringhe di note, ma solo i nomi B$ e A$ che le rappresentano, preceduti da una **x.** Il linguaggio si e' fatto quindi piu' stringato... e sintetico.

Fin qui non abbiamo fatto cenno al ritmo e alla durata delle note: in effetti il piccolo programma **cum** seguente suona l'inizio della **cumparsita**, ma di tango non c'e' traccia.

```
10 '....cum(parsita)
20 PLAY "o3c#dc#cc#c#bg#fc#
30 PLAY "c#dc#cc#c#
40 PLAY "o4c#o3af#c#
```

Occorrono quindi comandi che consentano di regolare il rapporto temporale tra nota e nota, nell'ambito di una certa velocita' di esecuzione del pezzo. Della durata delle note tratta il programma **via872** (figura 8.11). Alla linea 60 vediamo una PLAY che contiene una stringa di simboli per la generazione di quattro note (b, f#, g#, d#).

Come si puo' notare la stringa contiene anche dei numeri e dei punti piazzati subito dopo ogni singola nota. Nella sezione (a) della stessa figura 8.11 si riportano le equivalenti notazioni in una battuta sul pentagramma.

```
b4.p4.f#8.g#16.d#4.
b4.f#32p32g#32p32d#2

pausa (1...64) = 1/13


Premere un tasto per continuare
    <r> per ricominciare
```

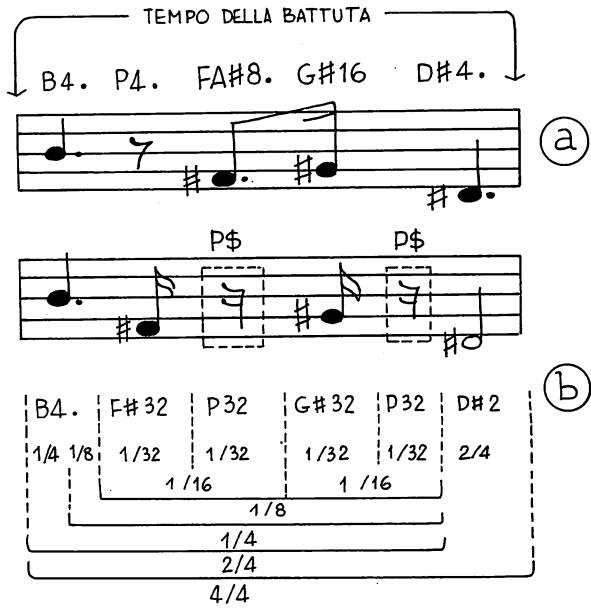
```
10 'via873 : .......... studio di pause
20 CLS:KEY OFF
30 SCREEN 1
40 LOCATE 10,10
50 PRINT"b4.p4.f#8.g#16.d#4."
60 PLAY"b4.p4.f#8.g#16.d#4."
70 LOCATE 12,10
80 PRINT"b4.f#32p32g#32p32d#2"
90 PLAY "b4.f#32p32g#32p32d#2"
100 PRINT
110 LOCATE 15,10
120 INPUT "pausa(1...64)=1/",P1$ '....13
130 'IF P1$="r" THEN 20
140 IF VAL(P1$)<1 OR VAL(P1$)>64 THEN 200
150 P$="p"+P1$
160 PLAY "b4.f#32;xp$;g#32;xp$;d#2"
170 LOCATE 22,10
180 PRINT "Un tasto per continuare "
181 LOCATE 23,10
182 PRINT "<i> per ricominciare";
183 QQ$=INPUT $(1)
185 IF QQ$="i" THEN 20
190 LOCATE 22,1:PRINT STRING$(85,32);
200 LOCATE 15,29:PRINT"              ";
205 LOCATE 16,1:PRINT STRING$(40,32);
210 GOTO 110
```


Figura 8.11

Cosi' il numero **4** dopo il simbolo della nota **b** significa che il **si** e' una semiminima (cioe' dura 1/4 del tempo assegnato a una battuta); il **punto** dopo il **4** [**4.**] significa che il **si** dura [1/4+1/8].
La notazione **p** sta per pausa e anche per essa valgono le nozioni di durata prima definite; cosi' p32 sta per pausa lunga 1/32 del tempo di battuta [4/4]. Nella sezione (b) della figura 8.11 e' rappresentata la battuta musicale corrispondente alla PLAY della linea 90.
Come al solito, volendo rendere parametrica l'istruzione, costruiamo una PLAY variabile a livello pausa. Il meccanismo e' costituito da una combinazione di due stringhe "p"+p1$ che definiamo p$, quest'ultima inserita con una **x** che la precede nella PLAY di linea 160.
Facendo girare il programma, quando si voglia assegnare valori diversi alla pausa p$, basta sceglierne uno compreso tra 1 e 64; battendo solo il **return** (linea 130) si torna daccapo.
Nello specchietto di figura 8.12 riportiamo alcune nozioni di solfeggio musicale.

Figura 8.12

La notazione del punto e' ricorsiva, in quanto allunga sempre della meta' la durata espressa dalle notazioni temporali che lo precedono. Cosi':

b. dura $\frac{4}{4}+\frac{2}{4}=\frac{6}{4}=\frac{3}{2}$

b.. dura $\{\frac{3}{2}+[\frac{1}{2}\cdot\frac{3}{2}]\}=[\frac{3}{2}+\frac{3}{4}]=\frac{12+6}{8}=\frac{18}{8}=\frac{9}{4}$

b... dura $[\frac{3}{2}\cdot\frac{3}{2}\cdot\frac{3}{2}]=\frac{27}{8}$ etc.

Un modo globale di definire la durata delle note e di premettere alla stringa la notazione Ln dove **n** (da 1 a 64) stabilisce la frazione di battuta che verra' assegnata a tutte le note che vengono scritte dopo, salvo diverse indicazioni locali. Naturalmente occorre un comando per regolare la velocita' di esecuzione del motivo musicale: e' quanto realizza il comando Tn dove **n** (da 32 a 255) definisce il numero di quarti di battuta (o semiminime) al minuto con il quale il pezzo viene eseguito. Il programma **tmus4** consente di sperimentare il sottocomando **Tn**, insieme a quello per la durata della nota: cosi', a parita' di valore della nota (semiminima, croma, biscroma, ...), s'impara a mettere tale valore in relazione al tempo di esecuzione (32=lentissimo; 255=allegro).

```
10 'tmus4:                    ........taratura del tempo musicale
15 CLS:KEY OFF:SCREEN 1
20 INPUT "SM/1'(32...255)=",T1$ '..n. di semiminime al minuto
25 IF VAL(T1$)<32 OR VAL(T1$)>255 THEN 20
30 INPUT "DUR.(1...64)=",W$  'durata della nota (32=biscroma)
35 IF VAL(W$)<1 OR VAL(W$)>64 THEN 30
40 T$="T"+T1$ '              sintesi della stringa tempo musicale
50 PLAY "xT$;" '             .......attuazione del tempo musicale
55 PRINT
60 R1$=INKEY$
65 IF R1$="r" THEN PRINT:GOTO 20
70 IF R1$="" THEN 60
80 IF ASC(R1$)<97 OR ASC(R1$)>103 THEN 60
85 R$=R1$+W$   '      sintesi della stringa durata della nota
90 PLAY "xr$;"
100 PRINT R$+" ";
110 GOTO 60
Ok
RUN
SM/1'(32...255)=255
DUR.(1...64)=32

c32 d32 c32 d32 c32 d32 c32 d32 c32 d32
e32 f32 e32 f32 e32 f32 e32 f32 e32 f32
g32 a32 g32 a32 g32 a32 g32 a32 g32 a32
b32 c32 e32 g32 c32
```

La sintassi della PLAY alla linea 50 e' studiata per ospitare il nome (T$) della variabile che pilota il ritmo, mentre quella della linea 90 regola la durata della nota. Le scelte del ritmo e della durata sono fatte alle linee 20 e 85, rispettivamente. Come si vede da una passata del programma, la print alla linea 100 provvede a scrivere un'eco su video della nota battuta alla linea 60 con relativa notazione temporale (32=biscroma).

Il successivo programma **tmus5** consente di sperimentare, interattivamente, insieme al ritmo e alla durata della nota, anche la localizzazione dell'ottava.

```
10 'tmus5:.....Taratura del tempo musicale
11 CLS:KEY OFF:SCREEN 2
20 INPUT "SM/1'(32...255)=",T1$ '(da 32 a 255):  n. di semiminime al minut
30 INPUT "DUR.(1...64)=",W$     '(da 1 a 64):durata della nota (32=biscrom
35 INPUT "n.Ottava(0...6)=",OC1$
36 OC$="0"+OC1$
37 PLAY "xoc$;"'                .......attuazione dell'ottava
40 T$="T"+T1$ '                 .......sintesi della stringa tempo musicale
50 PLAY "xT$;"'                 .......attuazione del tempo musicale
60 R1$=INKEY$
65 IF R1$="r" THEN PRINT:PRINT:GOTO 35
70 IF R1$="" THEN 60
80 IF ASC(R1$)<97 OR ASC(R1$)>103 THEN 60'
85 R$=R1$+W$'                   .......sintesi della stringa (nota e durata)
90 PLAY "xr$;"
100 PRINT OC$+R$+" ";'          .......visualizzazione di ottava, nota e dur
110 GOTO 60
Ok
RUN
SM/1'(32...255)=100
DUR.(1...64)=16
n.Ottava(0...6)=1
01a16 01b16 01c16 01d16
```

Col successivo programma **tmus6** la nota emessa automaticamente e' sempre la stessa (re diesis) e, dopo la scelta dell'ottava, del ritmo e della durata della nota, si puo' verificare con maggiore efficacia l'interdipendenza tra il ritmo (quarti al minuto) e durata della nota. Si noti che se il sottocomando T viene omesso la macchina assume implicitamente il valore di 120 quarti al minuto.

```
10 'tmus6:.....Taratura del tempo musicale
11 CLS:KEY OFF:SCREEN 2
15 INPUT "n.Ottava(0...6)=",OC1$
20 INPUT "SM/1'(32...255)=",T1$ '(da 32 a 255):  n. di semiminime al minut
30 INPUT "DUR.(1...64)=",W$     '(da 1 a 64):durata della nota (32=biscrom
36 OC$="0"+OC1$
37 PLAY "xoc$;"'                .......attuazione dell'ottava
40 T$="T"+T1$ '                 .......sintesi della stringa tempo musicale
50 PLAY "xT$;"'                 .......attuazione del tempo musicale
60 R2$=INKEY$:R1$="d#"
65 IF R2$="r" THEN PRINT:PRINT:GOTO 20
80 IF ASC(R1$)<97 OR ASC(R1$)>103 THEN 60'
85 R$=R1$+W$'                   .......sintesi della stringa (nota e durata)
90 PLAY "xr$;"
100 PRINT OC$+R$+" ";'          .......visualizzazione di ottava, nota e dur
110 GOTO 60
RUN
n.Ottava(0...6)=3
SM/1'(32...255)=60
DUR.(1...64)=16
03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#
03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#16 03d#
```

Un' ulteriore possibilita' di colorare il fraseggio musicale e' il sottocomando **Mf** dove **f** puo' assumere tre stati: **n**(=normale), **l**(=legato) e **s**(=staccato). Il sottocomando altera la durata della nota definita dal comando L(n) precedente. Il programma **suon1** fa ascoltare la stessa scala di do maggiore nei tre modi. Si noti che, mancando la definizione della durata nominale delle note, viene assunta implicitamente quella della semiminima.

```
10 'suon1
20 PLAY "o5t80"
30 PLAY "mncdefgab"
40 PLAY "mscdefgab"
50 PLAY "mlcdefgab"
```

8.7 UN LABORATORIO MUSICALE

Ci proponiamo ora di costruire uno strumento capace di visualizzare il rapporto tra ritmi, velocita' di esecuzione del pezzo, e valori delle note nel tempo di una battuta. Tutte le nozioni gia' acquisite e gli esperimenti fatti con i precedenti programmi conducono alla formulazione di un programma piu' completo per la logica di trattamento dei casi e piu' sofisticato come scenario di verifica. Si tratta del programma **walph33**, in cui una data battuta viene suonata indefinitamente, insieme con la sua rappresentazione grafica. In tale programma hanno un ruolo fondamentale soprattutto le variabili legate al ritmo e alle pause, mentre non sono possibili manipolazioni sulle altezze delle 3 note in gioco. Infatti l'altezza di tali note nella PLAY alla linea 380 viene stabilita, una volta per tutte, dalle relative assegnazioni alle linee 270 e 290. Possono essere, invece, variate esternamente la durata della prima nota (do di quarta ottava) e quella della successiva pausa. Di conseguenza, in funzione del numero di quarti per battuta prescelto (input di linea 160), le restanti due note (due mi di quarta ottava), risulteranno in cadenza ravvicinata col do della battuta successiva, soltanto se la somma delle durate dei quattro segni musicali (tre note e una pausa) e' sensibilmente diversa dalla durata della battuta.

Tra le scelte iniziali (figure 8.13 e 8.14) sono anche il ritmo di esecuzione espresso in numero di quarti (o semiminime) al minuto e il numero di battute che si desidera visualizzare. Durante l'esecuzione del pezzo, col tasto [a] si puo' selezionare il parametro che definisce la legatura della note, a parita' di tutti gli altri gia' impostati; oppure, premendo la barra spaziatrice, si puo' tornare all'inizio, dove viene posta la domanda sul tipo di risoluzione desiderata. Il programma puo' selezionare solo le due risoluzioni maggiori (screen 2 e 3). Sulla routine audiografica facciamo rilevare l'importanza delle due assegnazioni concettualmente piu' interessanti: quella che definisce lo spazio grafico (linea 2070) e quella che definisce il segmento equivalente alla durata di un quarto come unita' di misura per la rappresentazione dei valori musicali (linea 2080).

Figura 8.13

Figura 8.14

Segue **walph33**.

```
10 'walph33:valzer grafico
20 CLS:KEY OFF
30 INPUT "RISOLUZIONE=",RZ
40 IF RZ<2 THEN 30
50 CLS:KEY OFF:  SCREEN RZ
60 LOCATE 2,40
70 PR$="":PR1$="":PR2$=""
80 PRINT "****** W A L P H  33 ******"
90 LOCATE 2,1
100 INPUT "RITMO (Quarti al 1')=",R1$
110 IF RZ=2 THEN ZZ=2 ELSE ZZ=1
120 R$="T"+R1$ 'sintesi del s/comando per la 760
130 INPUT "DURATA PRIMA NOTA  1:",DN$
140 INPUT "DURATA DELLA PAUSA 1:",P1$
150 INPUT "N. QUARTI VISUALIZZ.=",QV
160 INPUT "N. QUARTI/BATTUTA   =",QB
170 '- - - - - - -> selezione del tipo di legatura
180 PRINT "(L)eg. (N)orm. (S)tacc.-->   "
190 J1$=INPUT$(1)
200 LOCATE 7,29:PRINT J1$
210 NS=ASC(J1$)
220 IF NS=108 OR NS=110 OR NS=115 THEN 230 ELSE 190
230 J$="M"+J1$   ' sintesi del s/comando per la 550
240 PLAY "xJ$;"
250 '- - - - - - - - - - - - - - - - - - - - - - -
260 GOSUB 2010
270 W1$="c"+DN$  'imposta il do di quarta ottava di
280 '             durata pari al valore DN (v. 360)
290 W$="e8e8"    'imposta due mi di quarta ottava di
300 '             entrambi di durata pari a 1/8.
310 P$="p"+P1$   'sintesi della pausa (da 1 a 1/64)
320 RES=QB/4-1/VAL(DN$)-1/VAL(P1$)-1/4
330 IF RES=0 THEN 380
340 IF RES>1 THEN PR$="p1"   :RES=RES-1
350 IF RES>.5 THEN PR1$="p2" :RES=RES-.5
360 IF RES>.25 THEN PR2$="p4":RES=RES-.25
370 IF RES>.125 THEN PR3$="p8":RES=RES-.125
380 PLAY"01;xR$;xw1$;xp$;xw$;xpr$;xpr1$;xpr2$;xpr3$;"
390 QQ$=INKEY$   'non e' sospensiva ma puo' filtrare
400 '  Filtro: introducendo "a" si puo'selezionare il
410 '          tipo di legatura tornando alla 825.
420 '--Cancellazione della rappresentazione grafica--
430 IF QQ$="a" THEN 440 ELSE GOTO 500
440 FOR I=8 TO 21
450 LOCATE I,1 :PRINT  STRING$(80,32)
460 NEXT
470 LOCATE 7,1:GOTO 180
480 '  Ripresa: introducendo uno spazio si ritorna da
490 '           capo alla linea 60.
500 IF QQ$=" " THEN 50 ELSE GOTO 270
2000 '- - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
2010 'routine audiografica
2020 '          Selezione del tipo di legatura.
2030 IF NS=108 THEN Z=1
2040 IF NS=115 THEN Z=3/4
2050 IF NS<>108 AND NS<>115 THEN Z=7/8
2060 DN=VAL(DN$)
2070 K=(639/QV)*QB
2080 J=4/QB
2090 VU=J*K/DN
2100 VO=VAL(P1$) 'durata della pausa
2110 VO=J*K/VO
2120 LINE (639,300/ZZ)-(0,200/ZZ),,B
2130 FOR I=0 TO 639 STEP 639/QV
2140 LINE (I,300/ZZ)-(I,200/ZZ)' marca il confine della battuta
2150 NEXT
2160 FOR I=0 TO 639 STEP K
2170 LINE (I,190/ZZ)-(I,308/ZZ)
2180 NEXT
2190 FOR I=0 TO 639 STEP K
2200 LINE (I,285/ZZ)-(I+VU*Z,300/ZZ),,BF 'prima nota: DO
2210 LINE (I+VU,300/ZZ)-(I+VU+VO,330/ZZ),,B '...la pausa
2220 NEXT
```

CAPITOLO
NOVE

LA GESTIONE DEGLI ARCHIVI

Nei capitoli precedenti abbiamo esaminato molti programmi per far conoscenza e pratica delle piu' importanti istruzioni Basic, nonche' delle situazioni di macchina e di ambiente in cui tali programmi erano ospiti.

Tutti questi programmi sono ben lungi dal creare un accumulo di dati cosi' cospicuo da saturare la memoria centrale. Sappiamo pero' che esistono le condizioni logiche per provocare una tale condizione: basta generare una matrice con un numero sufficientemente alto di elementi.

Ecco, quindi, l'utilita' di avere un altro tipo di memoria, molto piu' capace, in cui immagazzinare i dati. Ma come servirsene?

Chi di noi non possiede un'agenda sulla quale segna nome e numero di telefono di amici e conoscenti? L'utilita' di detta agenda e', pero', vincolata al corretto aggiornamento del suo contenuto.

L'amico che ha cambiato casa e numero di telefono, i nomi dei nuovi conoscenti o di quelli che non ci interessano piu', sono tutte operazioni connesse al suddetto aggiornamento.

Un archivio, in informatica, e' un insieme di dati (**file**) che ha impieghi piu' generali di quelli di una semplice agenda, ma, in linea di massima, le operazioni da compiere sono le stesse in entrambi i casi. In questo capitolo tratteremo le operazioni da compiere su M24 per gestire gli archivi, insieme ad alcuni esempi di utilizzo.

9.1 OPERAZIONI SU ARCHIVI

Finora abbiamo lavorato sui dischetti senza preoccuparci della struttura con cui i dati vengono archiviati; gli unici comandi Basic che conosciamo riguardano il **flusso** dei programmi tra dischetto e memoria centrale in occasione di salvataggio e richiamo di programmi. Piu' precisamente, ricordiamo:

SAVE"a:gamma	per salvare un programma di nome gamma
LOAD"a:delta	per caricare un programma di nome delta
RUN"a:delta	per caricare e eseguire un programma di nome delta
FILES"a:	per vedere l'elenco dei nomi dei **file**.

Tutti questi comandi interessano un particolare supporto fisico, il dischetto, identificato non come entita' in se', ma come drive che lo gestisce. E' come se, per nominare gli oggetti che possediamo, indicassimo la tasca nella quale sono contenuti: oggetti di tasca destra, oggetti di tasca sinistra.

Chiariamo, quindi, che i suddetti comandi possono anche riguardare il drive superiore, convenzionalmente indicato come il drive b. Cosi', se il dischetto e' quello superiore, i programmi gamma e delta possono essere salvati, richiamati eseguiti e elencati anche cosi':

```
SAVE"b:gamma
LOAD"b:delta
RUN"b:delta
FILES"b:
```

Ma un modo piu' semplice per cercare un'informazione sui dischetti e portarla in memoria deve potersi svincolare dall'unita' fisica, anche se cio' richiede la definizione di un minimo di convenzioni. E' cio' che normalmente si fa quando all'atto del caricamento iniziale del sistema operativo MS-DOS si definisce che il drive implicitamente assegnato dal sistema per le operazioni sui file e' il drive inferiore (drive a). Se invece si vuole cambiare convenzione e assumere che il drive implicito e' quello superiore allora e' necessario comunicarlo al sistema: da quel momento in poi, e fino a nuove istruzioni, il drive implicito (di default) sara' il drive b.

E' quindi possibile **ricercare un file attraverso il suo nome,** senza preoccuparci di indicare nel comando in quale drive la macchina deve andarlo a leggere. La ricerca ha esito positivo solo se il dischetto contenente il file desiderato e' inserito proprio nel drive che la macchina considera implicitamente selezionato.

Supponiamo, ad esempio di avere un contenitore di dischetti e di voler cercare il programma ITALIA semplicemente leggendo questo nome scritto a mano sull'etichetta. Una volta inserito tale dischetto in un drive, per caricare in memoria centrale il programma desiderato si imposta il comando [load"ITALIA]
Naturalmente, se non abbiamo montato il dischetto nel drive che la macchina considera come implicitamente selezionato in base alle convenzioni introdotte a inizio lavori, la richiesta non sara' soddisfatta e provochera' solo il messaggio FILE NOT FOUND (FILE NON TROVATO). A questo punto dovremo cambiar drive, oppure reimpostare il comando specificando anche il drive che il sistema deve selezionare.

9.2 GLI ARCHIVI DI DATI

Ora che sappiamo salvare e richiamare i programmi col loro nome e riferirci anche ai nomi dei volumi che li contengono, trattiamo delle modalita' per gestire un **archivio di dati** su cui far lavorare i programmi. Infatti il flusso dei dati, su cui lavorano i programmi fin qui esaminati, sono tutti **interni** e vanno persi appena spegniamo la macchina.

Vogliamo, dunque, crearci una banca dati esterna alla memoria centrale, residente su dischetto. Da qui si aprira' un **flusso esterno** di dati per portare il contenuto di una parte del dischetto in memoria, oppure il flusso inverso per generare un archivio e trasferirlo su dischetto.

Cominciamo con l'osservare che anche caricando o salvando un programma si crea un flusso di informazioni scambiate tra memoria

centrale e dischetti. Ma la differenza tra **file** (flussi) **programmi** e **file dati** sta proprio nell'ente che li utilizza. Mentre un file programma e' sotto la gestione del Basic che deve semplicemente caricare o scaricare un programma (vedremo che anche con i file programma potremo complicare la situazione con i comandi CHAIN e MERGE), un file di dati dipende soprattutto dal **tipo di file e di struttura.**

Esistono due tipi di file:

- file sequenziali,

- file random.

La diversita' consiste nel modo in cui sono strutturate le informazioni e quindi dal metodo di accesso per rintracciarle (operazione di lettura) o per registrarle (operazione di scrittura).

La registrazione sequenziale di un file, come dice il termine, si riferisce a una struttura di dati disposti nell'ordine in cui sono stati archiviati: in un file scritto in modo sequenziale le varie informazioni vengono reperite attraverso l'ordine con cui sono state registrate. Supponiamo, ad esempio, di voler costituire un file con i titoli dei libri di una biblioteca e col relativo nome dell'autore.

In informatica, secondo la terminologia anglosassone, l'insieme dei due elementi (titolo,autore) viene denominato **record,** mentre ciascuno degli elementi viene definito campo (**field**). In un file sequenziale, dunque, i record vengono scritti uno dopo l'altro e riletti nello stesso ordine con cui sono stati registrati. Le istruzioni Basic relative alla lettura di file sequenziali **leggono** sempre il file **dall'inizio,** anche se l'informazione desiderata si trova alla fine.

Un file scritto in modo **random,** invece, si riferisce alla possibilita' di accedere direttamente alle informazioni, senza tener conto dell'ordine in cui sono state registrate. Un file organizzato in modo random si dice anche, piu' propriamente, **diretto** e puo' essere letto sia in modo diretto, sia sequenziale.

9.3 ARCHIVI SEQUENZIALI

La prima operazione da fare e' quella di comunicare a M24 il tipo e il nome del file dati che intendiamo costituire. Questa definizione viene fatta con una particolare istruzione Basic, denominata **OPEN,** che, come dice il termine, **apre** un file. Il formato dell'istruzione e' il seguente:

OPEN "TA",NB,"F"

dove:

-F e' il nome del file con tutte le specificazioni osservate nel paragrafo precedente in tema di selezione di drive;

-NB e' il numero di buffer, o zona di memoria, dedicata al traffico dei dati scambiati col dischetto;

-TA specifica il tipo di accesso che puo' essere:

. random (TA= R)
. sequenziale in lettura (TA=I)
. sequenziale in registrazione (TA=0)
. sequenziale in aggiunta (Append:TA=A)

Vediamo ora il funzionamento della OPEN seguendo la figura 9.1 dove sono rappresentate tre zone di memoria corrispondenti a tre file interni di nome ITALIA, FRANCIA e GRECIA.

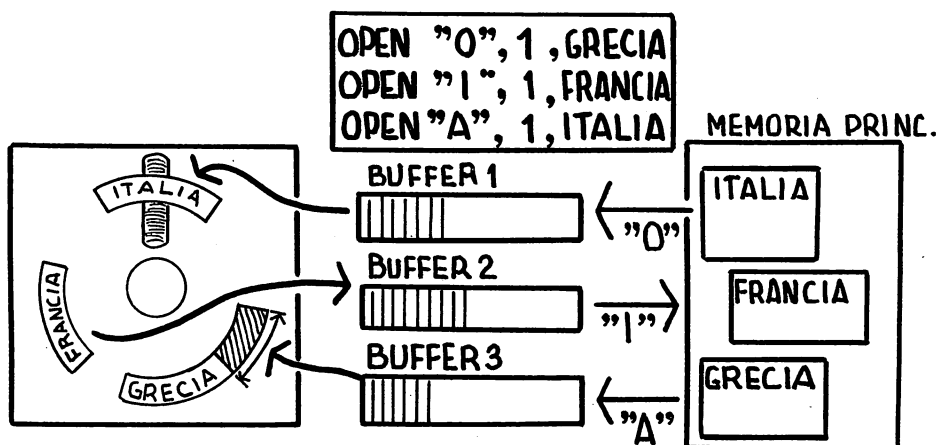



Figura 9.1

Con le tre OPEN elencate in un programma vengono riservati tre buffer di transito tra memoria e dischetto. In particolare, si noti che la OPEN"A" aggiunge i nuovi dati in coda al file gia' presente nel dischetto; se, invece, avessimo operato una OPEN"0",3,"ITALIA", il contenuto precedente sarebbe stato sostituito dal nuovo.

Nella gestione dei file occorre, inoltre, osservare che solitamente e' possibile disporre di tre zone buffer in modo da avere contemporaneamente aperti tre file. Se ne occorressero 15 (che e' il massimo su M24) basta al momento di richiamare il gwbasic impostare il numero di file. Il messaggio da scrivere e':

```
A> gwbasic  /f:15
```

Ovviamente, anche se l'abbinamento e' arbitrario, una volta associato un file a un numero di buffer, **finche' l'operazione e' attiva,** questo non puo' essere assegnato a un altro file.
Vediamo ora questo semplice programma, denominato **prores**, il primo di una serie di piccoli programmi per la gestione di un file.

```
10 'prores: istogrammi da vettore numerico estratto a caso
15 CLS    ' e successiva registrazione su file seq.(resvet)
17 KEY OFF:INPUT "RISOL.=",Z:SCREEN Z
20 INPUT "n.elem.=",N
30 DIM V(N)
40 INPUT "passo orizzont.=",P
46 T$=MID$(TIME$,7)
47 T=VAL(T$)
50 RANDOMIZE (Z+T)
60 INPUT;"  da <0> a < ",L:PRINT">"
70 FOR E=1 TO N
75 R=FIX((L+1)*RND):V(E)=R:PRINT V(E)" ";:NEXT E
80 PRINT
90 INPUT;"base istogr.=",B
100 FOR E=1 TO N
105 WINDOW (0,0)-(639,399)
110 LINE(100+(E-1)*P/N,10)-(100+B/N+(E-1)*P/N,10+V(E)),,B
120 NEXT E
121 A$=INPUT$(1)
123 Z=V(1)
130 OPEN "O",1,"resvet"
150 FOR E=1 TO N
160 PRINT# 1,V(E),
170 NEXT:CLOSE
```

In questo programma, il cui scopo e' riportato nella linea 10 dedicata al commento, osserviamo:

- l'istruzione INPUT alla linea 20, che chiede il numero degli elementi del vettore V generato con la RND e opportunamente stampato alla linea 75 (per maggiori dettagli sul trattamento delle liste si veda il paragrafo 9.4);

- il prelievo del primo elemento Z=V(1) e la sua inclusione con la variabile T nella Randomize non e' indispensabile;

- la linea 47 serve per convertire in numero la stringa T$, dato che l'argomento della Randomize deve essere un numero;

- la linea 46 estrae i caratteri dalla settima posizione in poi della stringa TIME$, che ,come sappiamo, e' costituita da **hh:mm:ss:** i caratteri **ss** occupano appunto la settima e l'ottava posizione;

- alla linea 130 la OPEN, con la "O" che sta per Output, apre un file su dischetto: non avendo dichiarato il nome del drive, ma solo il nome, s'intende che vogliamo riferirirci al drive di default (implicito per la macchina). Il nome prescelto e' **resvet.** Molti chiamano l'espressione "resvet" l'**identificatore del file**;

- la PRINT alla linea 160 scarica il flusso sul disco tramite il **buffer di transito** numero 1 e i dati vengono registrati in sequenza dall'inizio del file interno e nello stesso modo in cui apparirebbero sul video con l'analoga istruzione INPUT. Occorre fare, quindi, molta attenzione alla punteggiatura;

- la **close** alla 170 **chiude il file** aperto alla 130.

A questo punto e' bene accertarci se effettivamente un file di nome resvet esiste sul dischetto: serve allo scopo il comando FILES del BASIC, mediante il quale si possono controllare i nomi di tutti i file registrati sul dischetto di lavoro. Per quanto riguarda il contenuto del file prores appena registrato, occorre un opportuno programma che provveda a leggerlo.

Sembrera' strano, ma per leggere un file occorre prima aprirlo: la **OPEN di lettura** si deve quindi riferire a un file gia' aperto, nel senso di gia' creato ed esistente nel dischetto. Per leggere il file resvet, ad esempio, bisogna scrivere:

OPEN "I",1,resvet (oppure OPEN "I",1,**a:resvet**)

dove:

I sta per input verso la memoria centrale;

1 e' il buffer di transito;

resvet e' l'identificatore del file che vogliamo leggere.

Si noti che se scrivessimo **a:resvet** la lettura riguarderebbe comunque il drive **a** anche se il drive di default fosse il **b.**

Questa e' la OPEN che abbiamo scritto alla linea 30 del seguente programma **lesvet.**

```
10 'lesvet:lettura di un file sequenziale
20 DIM V(40)    '      di nome "resvet"
25 CLS:KEY OFF
30 OPEN "I",1,"a:resvet"
40 IF EOF (1) THEN END
50 FOR E=1 TO 35
60 INPUT# 1,V(E)
70 PRINT V(E);:NEXT:CLOSE
```

Su questa prima versione, evidentemente grossolana, e' utile fare la seguenti osservazioni.

1) Non e' sempre corretto leggere una quantita' costante di elementi (come appare nella FOR di linea 50 dove si alimenta per 35 volte la INPUT); ne' si puo' ricordare di quanti elementi e' costituito il file **resvet** ogni volta che viene registrato. Nella logica del programma, se gli elementi registrati sono piu' di 35 la lettura del file resvet non e' completa.

Se sono meno di 35, il file viene letto interamente, ma viene anche emesso un messaggio di errore "input past end", come si puo' sperimentare facendo girare il programma. Tale messaggio significa che si e' tentato di leggere piu' elementi di quanti ve ne fossero effettivamente contenuti.

2) L'istruzione alla linea 60 deve essere seguita dalla PRINT, in quanto la INPUT$ legge, senza visualizzarlo, il contenuto del buffer 1 associato al file reso disponibile dalla OPEN "I" di linea 30.

3) Il test in EOF (end of file=fine del file) messo all'esterno del ciclo FOR ... NEXT non serve. Messo all'interno, ad esempio tra la FOR e la INPUT, **sente la fine del file associato al buffer 1** e quindi puo' evitare l'inconveniente di far arrestare il programma quando la Input tenta di leggere piu' elementi di quanti ne contenga il file, come prima dicevamo.

Le prime riparazioni conducono alla seconda versione di lesvet, rinominata **lesevar**, riportata qui di seguito.

```
10 'lesevar:lettura di un file sequenziale di nome "resvet"
15 CLS:KEY OFF
20 INPUT;"n.elem.=",N: PRINT
30 DIM V(N)
40 OPEN "I",1,"a:resvet"
50 IF EOF (1) THEN END
60 FOR E=1 TO N
65 IF EOF (1) THEN END
70 INPUT# 1,V(E)
80 PRINT V(E);:NEXT:CLOSE:PRINT
```

Segue, qui di seguito, la registrazione di una serie di operazioni a partire dalla generazione di un vettore di otto elementi a carico del programma **prores** che li assegna alla variabile con indice V(E); questi otto valori vengono registrati sul dischetto come file di nome **resvet.**

```
LOAD"prores
Ok_
RUN
RISOL.=2
n.elem.=8
passo orizzont.=250
  da <0> a < 100>
 97   32   21   88   29   49   55   59
base istogr.=25
```

Successivamente, richiamato il programma **lesevar**, dopo aver dichiarato il numero degli elementi, si riottengono da dischetto gli stessi elementi.

Ulteriori idee suggeriscono di migliorare le prestazioni di lesevar sollevando il vincolo del nome del file congelato alla linea 40; nasce cosi' **lesevar1**, che porta anche la variante per quanto riguarda il numero degli elementi da leggere, grazie all'impiego, ora corretto, di EOF( ). Il programma lesevar1 e' cosi' articolato:

```
10 'lesevar1:programma per leggere un file
20 CLS:KEY OFF
30 INPUT;"nome testo-->",T$
40 PRINT : PRINT
50 OPEN "I",1,T$
60 IF EOF(1) THEN CLOSE:GOTO 20
70 LINE INPUT #1,V$(E): PRINT V$(E)
80 PRINT:PRINT "<"LEN(V$(E))">"
90 A$=INPUT$(1)
100 GOTO 60
```

Con la INPUT di linea 30 si introduce il nome del testo che viene assegnato alla variabile stringa T$, che compare anche nella OPEN di linea 50. La sospensiva di linea 90 serve per leggere passo a passo tutto il file. Un'informazione fornita da lesevar1, alla linea 80, ci ragguaglia sulla lunghezza di ogni elemento del file registrato.

Per **aggiungere qualcosa a un file sequenziale** gia' registrato, e' sufficiente riaprirlo in modo "A". Al programma prores gia' visto, basta aggiungere dalla linea 16 un gruppo di istruzioni con le quali si seleziona il tipo di operazione che si desidera attivare. Tali istruzioni sono:

```
16 INPUT "Nuova Registr.(N); Aggior.(A)-->  ",G$
17 IF G$="N" THEN M$="O" ELSE IF G$="A" THEN M$="A" ELSE
   GOTO 15
...
```

Naturalmente la 130 diventa:

```
130 OPEN M$,1,"a:resvet"
```

I nuovi elementi aggiunti, come sappiamo, vengono messi in coda a quelli esistenti nel file resvet, gia' registrato nel dischetto.

La nuova versione di prores, riportata di seguito, si chiama **ares**. Naturalmente, a questo nuovo programma e' possibile apportare tutte le migliorie gia' viste su prores, come l'apertura di un file di nome introdotto liberamente.

Per quanto riguarda il contenuto del file, come vedremo in seguito, anziche' introdurre un vettore generato dalla macchina, sara' interessante costruire delle routines che consentano di caricare su dischetto informazioni strutturate immesse da tastiera. La struttura puo' essere definita nell'ambito della routine di introduzione, come avviene nella generazione di una tabella a carico di particolari accorgimenti adottati intorno a istruzioni tipo INPUT; oppure mediante l'impiego di file ad accesso diretto, come vedremo al paragrafo 9.6, che, per loro natura, forniscono l'organizzazione piu' adatta ad ogni tipo di applicazione.

```
10 'ares: istogrammi da vettore num. casuale e successiva registra-
15 CLS  ' zione su file seq.(resvet). (N)=nuova; (A)=aggiornamento.
16 KEY OFF:INPUT "Nuova Registr.(N); Aggior.(A)-->  ",G$
17 IF G$="N" THEN M$="O" ELSE IF G$="A" THEN M$="A" ELSE GOTO 15
20 INPUT "n.elem.=",N
30 DIM V(N)
40 INPUT "passo orizzont.=",P
46 T$=MID$(TIME$,7)
47 T=VAL(T$)
50 RANDOMIZE (Z+T)
60 INPUT;"  da <0> a < ",L:PRINT">"
70 FOR E=1 TO N
75 R=FIX((L+1)*RND):V(E)=R:PRINT V(E)" ";:NEXT E
80 PRINT
90 INPUT;"base istogr.=",B
100 FOR E=1 TO N
105 WINDOW (0,0)-(639,399)
110 LINE(100+(E-1)*P/N,10)-(100+B/N+(E-1)*P/N,10+V(E)),,B
120 NEXT E
121 A$=INPUT$(1)
123 Z=V(1)
130 OPEN M$,1,"a:resvet"
150 FOR E=1 TO N
160 PRINT# 1,V(E),
170 NEXT:CLOSE
```

Infine, nella figura 9.2, riportiamo una sintesi illustrata dei programmi descritti in questo paragrafo.

Figura 9.2

9.4 LE TABELLE ELETTRONICHE

Quello che abbiamo finora sulle variabili e' che possono avere un nome a cui far corrispondere una grandezza, la quale a sua volta puo' assumere diversi valori. Abbiamo anche imparato a considerare una variabile come un contenitore al quale dare un nome e nel quale introdurre delle quantita' il cui valore non e' necessariamente stabilito in anticipo.

Per comodita' spesso il nome delle variabili e' legato alla natura delle quantita', cui si associano. Cosi' possiamo chiamare P il numero delle pagine di un libro e associare ad esso una variabile di nome appunto P. A tale nome possiamo quindi riferirci durante i calcoli come a un deposito destinato a conservare, fino alla prossima modifica, il numero delle pagine di un libro.

Queste sono le cosiddette variabili semplici. Ma, come abbiamo gia' anticipato al paragrafo 6.3 in tema di classificazione di variabili, esistono anche quelle che, grazie a un indice che funge da suffisso a uno stesso simbolo, rappresentano un insieme di variabili semplici e alle quali va quindi associato un congruo numero di contenitori. Cosi', mentre con la lettera P indichiamo semplicemente il numero delle pagine, ad esempio, di un libro, con P(1) -si legge pi uno- posso indicare il numero delle pagine del primo capitolo di un libro, con P(2) quello delle pagine del secondo capitolo,con P(N) il numero delle pagine dell'ennesimo capitolo.

Oltre a una lista di variabili P(1),P(2)...P(N), con un solo indice, che, ricordiamo, si chiama anche vettore, esistono anche variabili con piu' di un indice.

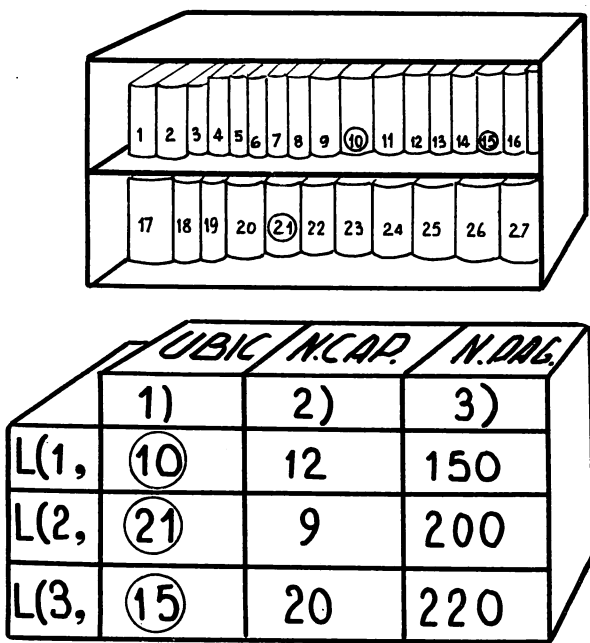



Figura 9.3

Ad esempio, se L e' una variabile associata al numero di libri di una biblioteca e vogliamo costruire una tabella formata da tante righe R quanti sono i libri e tante colonne C quanti sono gli attributi che vogliamo evidenziare di tali libri (soggetto, autore, editore..etc.), avremo un insieme di contenitori in numero pari a R X C.

Nella tabella di figura 9.3 sono stati scritti nove numeri che si riferiscono a tre libri contenuti in uno scaffale come quello disegnato. Per ciascono di questi tre libri esiste una riga della tabella che indica l'ubicazione del libro nello scaffale, il numero dei capitoli di cui il libro e' costituito e il numero totale delle pagine.

L'entita' libro si e' cosi' trasformata nell'entita' riga. Ed alcuni attributi del libro come l'ubicazione, il numero dei capitoli e il totale delle pagine, compaiono come elementi di quella riga. Ad esempio, il libro in prima riga e' quello che occupa la posizione 10 e che contiene 12 capitoli, per un totale di 150 pagine. Le stesse cose vengono lette nella tabella. Si e' cosi' assegnato un significato agli elementi che compaiono nella stessa riga e uno a quelli che occupano lo stesso ordine di colonna.

Ricordiamo, inoltre, che per denominare un elemento della tabella -nel caso specifico si tratta di una **matrice quadrata**- si usa una notazione con due indici: uno per le righe, l'altro per le colonne. In tal modo, ad esempio, gli elementi della riga 2 saranno L(2,1), L(2,2), L(2,3), che si leggono elle due uno, elle due due, elle due tre.

Come tutte le altre **variabili**, anche quelle **con indice** si distinguono per qualita' di contenitore: esistono, dunque, variabili abilitate ai numeri interi, ai numeri reali in semplice e doppia precisione e quelle destinate alle stringhe. Nell'ordine, i simboli da mettere in coda al nome sono: % , ! , # , $.

Come per le variabili semplici, se viene omesso il simbolo, s'intende la variabile numerica in semplice precisione. Nell'esempio dei libri si puo' quindi scrivere L%(R,C), L(R,C), L#(R,C), L$(R,C); si noti che, anche se il nome e' lo stesso (L), si tratta di variabili diverse perche' non appartengono allo stesso tipo.

Poiche' le variabili con indice possono contenere un numero di elementi qualsiasi, occorre definire la **dimensione** massima di tali elementi. Tale definizione viene fatta con la frase **DIM** che puo' definire anche piu' di una variabile. Ad esempio con la

```
DIM S(15),C$(24,2)
```

si e' definito che la variabile numerica S e' costituita da una lista che puo' contenere al massimo 16 elementi numerati da 0 a 15, registrati in semplice precisione. Per la variabile C$ si tratta di una tabella a due entrate (o matrice a due dimensioni) che puo' contenere al massimo 75 stringhe di caratteri alfanumerici individuate dai numeri di riga e di colonna i cui valori massimi sono rispettivamente 24 e 2. Avrete notato che il valore minimo implicito di ogni dimensione e' zero. Volendo numerare gli elementi a partire da 1, occorre usare anche l'istruzione **OPTION BASE 1** che serve appunto a tale scopo. Il valore implicito massimo assegnato all'indice, in assenza della DIM, e' pari a 11.

Cosi' il numero degli elementi di una matrice rettangolare, per la

quale non sia stata specificata la dimensione massima, potra' contenere un massimo di 11x11=121 elementi.

Vediamo ora come consolidare le nozioni acquisite sulle variabili con indice attraverso alcuni piccoli programmi: in essi, ovviamente, vengono impiegate le stesse istruzioni che abbiamo imparato a conoscere con le variabili semplici, ma occorre fare maggiore attenzione per la presenza degli indici. Ad esempio, per assegnare un valore a una variabile con indice occorre, nella relazione di assegnazione, specificare esattamente gli indici, insieme al nome della variabile. Nel caso della matrice di libri trattata all'inizio, porre L(3,2)=20 significa, in particolare, assegnare 20 capitoli al libro numero 3. E ancora, scrivere la relazione P(4)=35 significa assegnare 35 pagine al capitolo 4. In fondo una tabella e' un insieme di variabili con indice, ciascuna delle quali puo' essere usata come se fosse una variabile semplice. Scrivere tante frasi di assegnazione quanti sono gli elementi di una lista o di una tabella e' uno dei procedimenti per costruire appunto una lista o una tabella.

La tabella di figura 8.4 puo' quindi essere costruita con le seguenti istruzioni di programma:

```
L(1,1)=10:L(1,2)=12:L(1,3)=150:
L(2,1)=21:L(2,2)=9: L(2,3)=200:
L(3,1)=15:L(3,2)=20:L(3,3)=220
```

Lo stesso risultato si poteva ottenere con la coppia di frasi READ/DATA nell'esempio qui sotto riportato.

```
READ L(1,1),L(1,2),L(1,3),L(2,1),L(2,2),L(2,3),L(3,1),L(3,2),L(3,3)
DATA 10,12,150,21,9,200,15,20,220
```

o, ancora, con una READ nel corpo di una coppia di cicli FOR...NEXT, uno dentro l'altro:

```
OPTION BASE 1
FOR R=1 TO 3
FOR C=1 TO 3
READ L(R,C)
NEXT C
NEXT R
DATA 10,12,150,21,9,200,15,20,220
```

Un programma equivalente e':

```
OPTION BASE 1
FOR C=1 to 3
FOR R=1 TO 3
READ L(,C)
NEXT R
NEXT C
DATA 10,21,15,12,9,20,150,200,220
```

La differenza consiste nel fatto che mentre il primo ha il nucleo interno che cicla sulle colonne, il secondo ha il nucleo interno che

cicla sulle righe: se vogliamo costruire la stessa tabella, la sequenza dei valori nella frase DATA sara' quindi diversa. Per maggiori dettagli sulle frasi DATA/READ si veda il paragrafo 3.1.

Un metodo interattivo per il caricamento iniziale di una lista o di una tabella e' tuttavia quello che impiega la frase INPUT come mezzo di introduzione degli elementi da tastiera. Vediamone qualche esempio nei seguenti programmi **lislo1** e **tablo.**

```
10 'lislo 1  :caricamento di una lista di 5 elem.
15 CLS:KEY OFF
20 FOR E=1 TO 5
30 INPUT V%(E)
40 NEXT E
45 FOR E=1 TO 5
50 PRINT "  V("E")=";V%(E)
55 NEXT E
60 END
RUN
? 12
? 5
? 25
? 3
? 65
  V( 1 )= 12
  V( 2 )= 5
  V( 3 )= 25
  V( 4 )= 3
  V( 5 )= 65
Ok
```

```
10 'tablo:caricamento di una tabella 5 X 3
15 CLS:KEY OFF
20 FOR R=1 TO 5
30 FOR C=1 TO 3
40 INPUT L$(R,C)
50 NEXT C
60 NEXT R
70 FOR R=1 TO 5
80 FOR C=1 TO 3
90 PRINT " L("R","C")=";L$(R,C)
100 NEXT C
110 NEXT R
120 END
RUN
? Nel
? mezzo
? del
? cammin
? di
? nostra
? vita
? mi
? ritrovai
? per
? una
? selva
? oscura
? che'
? la
 L( 1 , 1 )=Nel
 L( 1 , 2 )=mezzo
 L( 1 , 3 )=del
 L( 2 , 1 )=cammin
 L( 2 , 2 )=di
 L( 2 , 3 )=nostra
 L( 3 , 1 )=vita
 L( 3 , 2 )=mi
 L( 3 , 3 )=ritrovai
 L( 4 , 1 )=per
 L( 4 , 2 )=una
 L( 4 , 3 )=selva
 L( 5 , 1 )=oscura
 L( 5 , 2 )=che'
 L( 5 , 3 )=la
Ok_
```

Col programma lislo1 si crea una lista di nome V% costituita da 5 elementi numerici interi, nell'ordine in cui si presentano in figura 8.5; con il programma tablo, invece, si carica una tabella a due entrate di elementi alfanumerici con 5 righe e 3 colonne (vedi ancora la figura 9.4).

V%(1)	V%(2)	V%(3)	V%(4)	V%(5)
12	5	25	3	65

C / R	1)	2)	3)
V (1,	Nel	mezzo	del
V (2,	cammin	di	nostra
V (3,	vita	mi	ritrovai
V (4	per	una	selva
V (5	oscura	che'	la

Figura 9.4

La procedura di **caricamento del vettore** V e' composta di una frase INPUT alla 30 nel corpo di un ciclo FOR...NEXT in cui l'indice di controllo del ciclo coincide con l'indice E del vettore; quella della tabella e' composta di una frase INPUT nel corpo di un ciclo FOR...NEXT che evolve sulle colonne e di un ciclo piu' esterno che cicla sulle righe.

Impostando il comando RUN, in entrambi i casi, i programmi si arrestano alla frase INPUT (frase 30 nel primo e 40 nel secondo) in attesa che l'operatore introduca i dati. Il messaggio di attesa, come e' stato spiegato al paragrafo 3.1, e' un punto interrogativo (?). Poiche' la frase INPUT, in lislo1, e' nel corpo di una FOR...NEXT da 5 cicli, M24 chiedera' per 5 volte all'operatore di introdurre dati. Nel programma tablo la frase INPUT chiede, invece, di essere soddisfatta per 15 volte.

Infine, la frase PRINT dara' conferma dei dati caricati visualizzando i valori introdotti, assegnandoli contemporaneamente ai diversi elementi della lista o della tabella. Anche per la PRINT impieghiamo dei cicli FOR...NEXT per gestire gli indici.

Al programma tablo apportiamo ora alcune modifiche che riguardano sia l'introduzione dei dati o l'emissione dei risultati, sia la loro visualizzazione.

Piu' precisamente, abbiamo introdotto due INPUT per la definizione, a scelta dell'operatore, della dimensione della tabella: la linea 20 chiama RT il numero delle righe totali, mentre la linea 40 quello delle colonne. Al momento dell'utilizzazione, occorre fare attenzione che, in assenza di specifiche istruzioni (DIM), la capacita' di memoria assegnata dalla macchina alla matrice e' solo di 11 x 11 elementi.

La modifica che riguarda il modo di visualizzare i dati introdotti e' stata operata aggiungendo un punto e virgola nella INPUT di caricamento della V e introducendo una PRINT alla linea 95, quando il programma ha finito di trattare una riga di tabella e sta per iniziare il trattamento della successiva. L'effetto prodotto in tal modo e' proprio la visualizzazione di una riga della tabella alla quale viene riservata un'intera riga di video o di stampa.

Analoga modifica per la visualizzazione dei risultati si ottiene con una PRINT alla linea 155. Il programma cosi' corretto e rinominato **tablo3**, in esecuzione con dei dati di prova , ha prodotto i seguenti risultati.

```
10 'tablo3:caricamento tabella R X C
15 CLS:KEY OFF
20 INPUT "n.righe tot.=",RT 'attenzione a non superare il valore
30                          'massimo=10 dell'autodimensionamento
40 INPUT "n.colonne tot.=",CT
50 FOR R=1 TO RT              'nota che le variab. di controllo
60 FOR C=1 TO CT              'dei cicli R e C sono diverse dai
70 PRINT "  V("R","C")=";        'limiti RT e CT.
80 INPUT; "",V(R,C)            'le linee 70 e 80 sono unite dal ;
90 NEXT C                     'nella 70 tutto cio' che sta tra
95 PRINT
100 NEXT R                    'due apici (") consecutivi viene
105 PRINT
110 '**STAMPA TABELLA**        stampato cosi' com'e'.
120 FOR R=1 TO RT
130 FOR C=1 TO CT
140 PRINT "*V("R","C")=" V(R,C); 'nota ancora il ruolo degli (")
150 NEXT C
155 PRINT
160 NEXT R
170 END
RUN
n.righe tot.=3
n.colonne tot.=3
  V( 1 , 1 )=45  V( 1 , 2 )=12  V( 1 , 3 )=14
  V( 2 , 1 )=11  V( 2 , 2 )=54  V( 2 , 3 )=85
  V( 3 , 1 )=14  V( 3 , 2 )=21  V( 3 , 3 )=32

*V( 1 , 1 )= 45 *V( 1 , 2 )= 12 *V( 1 , 3 )= 14
*V( 2 , 1 )= 11 *V( 2 , 2 )= 54 *V( 2 , 3 )= 85
*V( 3 , 1 )= 14 *V( 3 , 2 )= 21 *V( 3 , 3 )= 32
Ok
```

La seguente variante del programma tablo3, ridenominato **restab1**, consente di trasferire una matrice di dati su disco.

```
10 'restab1 (ex tablo3):........caricamento di una tabella R X C
20 CLS:KEY OFF
30 INPUT "n.righe tot.=",RT 'Attenzione a non superare il valore
40                          'massimo=10 dell'autodimensionamento.
50 INPUT "n.colonne tot.=",CT
60 FOR R=1 TO RT              'Nota che le variab. di controllo
70 FOR C=1 TO CT              'dei cicli R e C sono diverse dai
80 PRINT "  V("R","C")=";     'limiti RT e CT.
90 INPUT; "",V(R,C)           'Le linee 70 e 80 sono unite dal ;
100 NEXT C                    'nella 80 tutto cio' che sta tra
110 PRINT                     'due apici (") consecutivi viene
120 NEXT R                    'stampato cosi' com'e'.
130 PRINT
140 '**STAMPA TABELLA**
150 FOR R=1 TO RT
160 FOR C=1 TO CT
170 PRINT "*V("R","C")=" V(R,C); 'nota ancora il ruolo degli (")
180 NEXT C
190 PRINT
200 NEXT R
210 OPEN"O",1,"tab"           'si apre un file di nome "tab"
220 PRINT#1,RT:PRINT#1,CT     'si registrano le dim. di "tab"
230 FOR R=1 TO RT
232 FOR C=1 TO CT
234 PRINT#1,V(R,C)            'si registrano gli elem. di "tab"
236 NEXT:NEXT::CLOSE          'si chiude il file
240 END
RUN
```

Si noti (linea 220) che i primi elementi registrati su dischetto sono RT e CT, cioe' i numeri di riga e di colonna. I successivi elementi trasferiti sono sotto il controllo di un doppio ciclo che e' regolato proprio dalle dimensioni della matrice: in tal modo la **print** scrive su disco una sequenza di dati che rispetta la struttura della matrice stessa.

Per vedere il contenuto del file "tab" si puo' lanciare il comando BASIC: SHELL "TYPE tab [CR]. Supponiamo ad esempio di aver registrato una tabella di 2 righe e 3 colonne (figura 9.5). Ecco cosa vedremo in risposta al comando shell:

2 3 12 13 14 15 16 17

R \ C	1	2	(3)
1	12	13	14
(2)	15	16	17

Figura 9.5

Ma si puo' anche scrivere un programma per la lettura di un file sequenziale che ripeta in modo duale il processo gia' messo a punto per restab1. E' cio' che realizza il seguente programma **lestab** per la lettura di una tabella registrata su dischetto:

```
10 'lestab:lettura di una tabella su f.seq.
15 CLS:KEY OFF
20 OPEN "I",1,"tab"
30 INPUT#1,RT:INPUT#1,CT
40 FOR R=1 TO RT
50 FOR C=1 TO CT
60 INPUT#1,V(R,C)
65 PRINT V(R,C);
70 NEXT:PRINT:NEXT:CLOSE
```

In **lestab** la line 30 legge le dimensioni della tabella espressa nei due primi valori (primi 2 elementi del file "tab") da RT e CT chè impiega nelle due FOR...NEXT. La OPEN alla linea 20 apre un flusso, estrendo dal file "tab" un elemento alla volta e trasferendolo in memoria centrale, tramite il buffer 1, in un file interno. Il primo Input in 30 estrae il primo elemento (5) e lo assegna alla variabile RT (5 righe); il secondo input nella stessa 30 estrae il secondo elemento (3) e lo assegna alla variabile alla variabile CT (3 colonne). I successivi elementi del file sono estratti dalla input alla linea 60 che e' piazzata nel corpo di una doppia FOR...NEXT, regolata nello strato piu' esterno da RT e in quello piu' interno da CT. I vari elementi sono assegnati al vettore bidimensionale V(R,C), che li accoglie in una struttura matriciale che conserva il valore di posizione delle righe e delle colonne.

In lestab2 presentiamo le prime e immediate migliorie apportate a lestab: si tratta della scelta del nome del file operata alla linea 14.

```
10 'lestab2:lettura di una tabella su f.seq.
12 CLS:KEY OFF
14 INPUT;"nome file--> ",F$
16 PRINT
20 OPEN "I",1,F$
30 INPUT#1,RT:INPUT#1,CT
40 FOR R=1 TO RT
50 FOR C=1 TO CT
60 INPUT#1,V(R,C)
65 PRINT V(R,C);
70 NEXT:PRINT:NEXT:CLOSE
```

Nel successivo programma **lestab4** riportiamo i primi tentativi di trattamento degli elementi di una tabella.

```
10 'lestab4:                  lettura di una tabella su f.seq.
12 CLS                        'e calcolo della media per riga.
14 INPUT;"nome file--> ",F$
16 PRINT
20 OPEN "I",1,F$
```

```
30 INPUT#1,RT:INPUT#1,CT'     estrae gli elementi
40 FOR R=1 TO RT             'dalla tabella R X C
50 FOR C=1 TO CT
60 INPUT#1,V(R,C)
70 NEXT:NEXT:CLOSE
75 FOR R=1 TO RT
80 FOR C=1 TO CT
85 PRINT V(R,C);             'stampa gli elementi
90 NEXT:PRINT:NEXT           'dalla tabella R X C
95 PRINT
100 FOR R=1 TO RT
110 FOR C=1 TO CT
120 V1(R,C)=2*V(R,C)         'raddoppia gli elementi
130 PRINT V1(R,C);           'e crea la nuova tabella
140 NEXT:PRINT:NEXT
145 PRINT
150 FOR R=1 TO RT            'In 170 calcola  e in 190 stampa
160 FOR C=1 TO CT            'la media dei valori della riga;
170 S(R)=V(R,C)+S(R)         'La media viene calcolata in base
190 NEXT:PRINT S(R)/CT:NEXT 'al numero delle colonne CT.
```

Nel successivo programma **curtab** si fanno esperimenti di traslazione su video di tabelle registrate in un file interno.

```
5 'curtab:piazzare la tabella in ogni parte del video con LOCATE
10 '       deriva da restab (ex tablo3):caricamento tabella R X C
15 CLS:KEY OFF
17 INPUT"DOVE ? --> ",A,D
20 INPUT "n.righe tot.=",RT 'attenzione a non superare il valore
30                          'massimo=10 dell'autodimensionamento

40 INPUT "n.colonne tot.=",CT
50 FOR R=1 TO RT             'nota che le variab. di controllo
60 FOR C=1 TO CT             'dei cicli R e C sono diverse dai
70 PRINT "  V("R","C")=";    'limiti RT e CT.
80 INPUT; "",V(R,C)          'le linee 70 e 80 sono unite dal ;
90 NEXT C                    'nella 70 tutto cio' che sta tra
95 PRINT                     'due apici (") consecutivi viene
100 NEXT R                   'stampato cosi' com'e'.
105 PRINT
110 '**STAMPA TABELLA**
120 FOR R=1 TO RT
125 LOCATE A+R,D+C
130 FOR C=1 TO CT
140 PRINT " *V("R","C")=" V(R,C); 'nota ancora il ruolo degli (")
150 NEXT C
155 PRINT
160 NEXT R
170 END
```

9.5 UN PO' DI ORDINAMENTI

Abbiamo cosi' introdotto i concetti di liste e matrici e costruito strumenti software per caricarle in memoria. Il prossimo obiettivo e' quello di costruire un algoritmo per ordinare quelle liste.

Com'e' noto, saper mettere in ordine dei numeri costituisce una delle piu' comuni e piu' richieste routine che vengono assegnate a un elaboratore. Tra l'altro l'operazione di ordinamento e' anche una tra le piu' importanti relazioni matematiche. In informatica, interessa in particolar modo quella che si riferisce all'ordine stretto, ovvero, quello che consente, in un dato insieme di elementi, di poter sapere tra due elementi qualsiasi quale di essi viene prima dell'altro.

Esistono infiniti esempi di relazioni di ordinamento. Molti di essi sono tanto comuni che neppure ci accorgiamo quanto sono frutto di continue astrazioni. Esempio banale e' quello della lettura di un orario ferroviario che comporta molte operazioni di scelta e impercettibili processi di ordinamento, tanto che potremmo considerarlo un test di intelligenza. Infatti per arrivare all'informazione desiderata "...a che ora parte un dato treno per..." occorre spesso selezionare numerose classi di eventi, o condizioni logiche (solo rapidi prima classe...) oppure obiettivi alternativi (stazioni di fermata, coincidenze, instradamenti, prezzi...).

Negli ordinamenti e' importante innanzitutto decidere il criterio o i criteri che intervengono nel processo. Ad esempio, nei giochi d'ordine con i bambini, nell'impiego di materiale concreto strutturato (i famosi blocchi logici del Dienes) bisogna scegliere con quale criterio iniziale cominciare a dividere i blocchi. Se, come primo ordinamento, decidiamo di scegliere il criterio del colore, i successivi ordinamenti saranno tutti condizionati da quella scelta iniziale e cosi' via, proseguendo fino alla fine del processo, quando non potranno intervenire ulteriori criteri di selezione.

Un altro esempio di ordine stretto e' costituito dalla successione temporale di eventi, come l'estrazione di oggetti colorati da un'urna: prima e' uscito l'oggetto rosso, poi quello verde, poi tre gialli di seguito, etc. Come sappiamo, i concetti alla base del ragionamento sono tra i piu' semplici e tra i piu' intuitivi, ma, nello stesso tempo, di non facile formalizzazione.

D'altra parte, se vogliamo che l'operazione diventi tanto automatica da poterla assegnare a un'elaboratore, occorre saperla materializzare in un diagramma logico di tipo generale. Sulle tecniche di ordinamento e relativi metodi sono stati scritti interi trattati.

Il metodo che qui riportiamo e' uno dei piu' semplici e propedeutici: sfortunatamente ha l'inconveniente di essere lento nel produrre risultati quando il numero degli elementi da ordinare supera alcune centinaia.

Immaginiamo ora di avere un vettore V(E) costituito da N numeri identificati attraverso un indice, sara' quindi variabile da 1 a N. Per analizzare il processo di ordinamento, rappresentiamo un caso semplice costituito da un insieme di cinque numeri (N=5). Le disposizioni iniziale e finale siano quelle rappresentate nella figura 8.8.

Ci dovremo servire di alcuni artifici per raggiungere lo scopo. Uno di questi e' l'impiego di un **segnaposto** (E) per saper descrivere con

esattezza i vari passi della procedura e conoscere in ogni momento in quale punto ci troviamo.

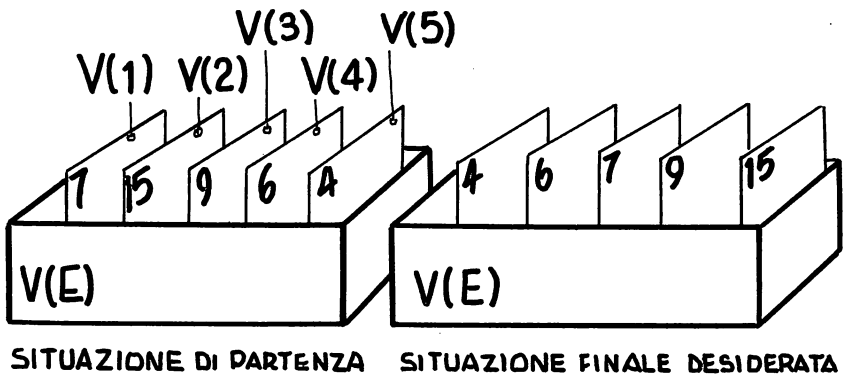



Figura 9.6

Osservando la situazione di partenza, il primo impulso sarebbe di prendere V(5)=4 e metterlo in prima posizione, ma dovremmo nello stesso tempo mettere il 7 al posto del 4. Per far questo dovremmo innanzitutto trovare tra tutti gli elementi quello che contiene il valore piu' basso (il 4). Si comprende presto che a ogni elemento piazzato al posto giusto si deve ripetere il processo per il successivo ancora da piazzare; ma ogni volta escludendo nei confronti tutti gli elementi gia' piazzati. In pratica, anche se si arriva comunque alla soluzione, questo procedimento comporta l'impiego di un maggior numero di artifici di quello che invece vi proponiamo.

Prendiamo ancora il caso semplice di 5 numeri da ordinare, gia' rappresentato in figura 9.6. Iniziamo a confrontare V(1) con V(2). Se V(1) e' minore o uguale a V(2) - come nel nostro caso - si passa a confrontare V(2) con V(3). Se V(2) fosse minore o uguale a V(3) e altrettanto accadesse per V(3) e V(4) e per V(4) e V(5), i numeri sarebbero gia' ordinati. Nell'esempio, invece, V(2) non e' minore o uguale a V(3). Allora scambiamo i contenuti per fare in modo che cio' accada. In tal modo, a inversione effettuata, la situazione si presenta come in figura 9.7.

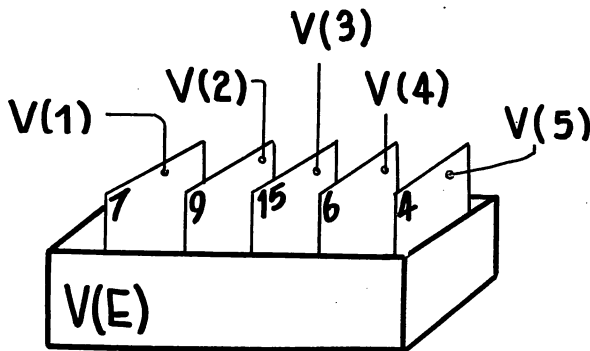
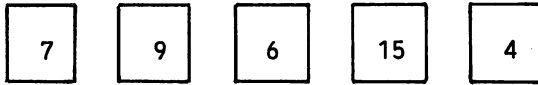



Figura 9.7

Eravamo rimasti al confronto tra V(2) e V(3) il cui esito ci ha indotto a scambiare i loro contenuti. Avanzando con gli ultimi due confronti, il primo da' luogo a un'altra **inversione dei contenuti,** perche' V(3) e' maggiore di v(4) e la situazione risultera':

Per l'ultimo confronto (il quarto) V(4) e' maggiore di V(5). Come prima, ancora un'inversione e alla fine la situazione dei **contenuti** rispetto alle **posizioni** degli elementi si presenta come in figura 9.8.

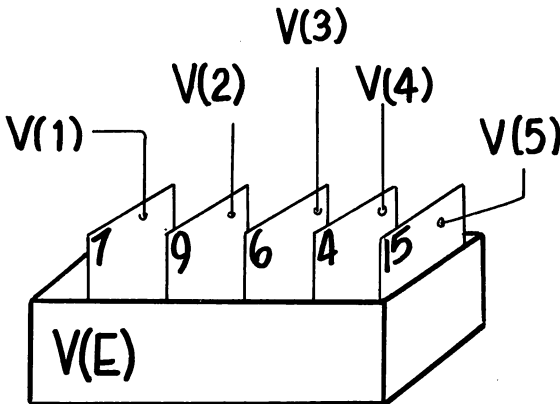



Figura 9.8

In questa prima serie di confronti, che chiameremo **passata,** su un insieme di cinque numeri sono stati eseguiti (5-1) confronti. Nella seconda passata ripetiamo esattamente quello che abbiamo fatto nella prima, partendo dalla situazione finale ereditata. Il primo confronto non porta a uno scambio tra V(1) e V(2); il secondo porta allo scambio tra V(2) e V(3), il terzo tra V(3) e V(4), mentre il quarto fallisce. La situazione alla terza passata conduce a uno scambio per i primi due confronti, mentre il terzo e il quarto sono improduttivi. Alla quarta passata il primo confronto porta all'inversione di V(2) con V(1), mentre gli ultimi tre sono inutili.

La figura 9.9, che illustra il processo logico descritto, consente di fare due interessanti constatazioni. La prima e' che nell'ambito di una passata non tutti i confronti sono produttivi ai fini di un'inversione. La seconda constatazione e' che se si giunge sicuramente al risultato con N-1 passate, puo' essere che, in qualche caso, ne bastino meno.

L'algoritmo, quindi, da realizzare e' quello che nella prima passata fa almeno 4 confronti, nella seconda 3, nella terza 2, mentre nella quarta ne basta 1.

In generale, nella P-esima passata bastano N-P confronti, dove N e' il numero degli elementi da ordinare e P e' il numero della passate.

PRIMA PASSATA				
7	15	9	6	4
7	9	15	6	4
7	9	6	15	4
7	9	6	4	15

SECONDA PASSATA				
7	9	6	4	15
7	6	9	4	15
7	6	4	9	15
7	6	4	9	15

TERZA PASSATA				
6	7	4	9	15
6	4	7	9	15
6	4	7	9	15
6	4	7	9	15

QUARTA PASSATA				
4	6	7	9	15
4	6	7	9	15
4	6	7	9	15
4	6	7	9	15

..... confronto che **non** ha dato luogo a inversione

----- configurazione risultante a seguito di confronto che ha dato luogo a inversione

Figura 9.9

Della possibilita' di poter ridurre il numero dei confronti in ogni passata non si terra' conto nella prima stesura del programma, denominato **NORDOT** (**OR**dinamento **N**on **OT**timizzato di N numeri). L'altra osservazione suggerisce di individuare una scorciatoia al metodo, nei casi in cui certe sequenze di numeri non richiedano tutte le passate. In effetti, riferendoci ancora alla figura 9.9, se la sequenza iniziale fosse quella all'inizio della terza passata (cioe' 6/7/4/9/15), dopo due passate i numeri sarebbero gia' ordinati e le ultime due passate sarebbero inutili.

Per risparmiare almeno la quarta (e quindi il tempo per fare i rispettivi confronti) nel programma verra' adottato l'artificio di far saltare la passata successiva a quella in cui neppure un confronto ha prodotto inversioni. Per chiarezza, neppure questa miglioria verra' adottata nella prima stesura del programma.

Il problema, a questo punto ci sembra sufficientemente analizzato per tentare di formulare la prima versione del meccanismo che lo risolve. Restano, naturalmente, da progettare i dettagli interni alla procedura, tra cui quello della gestione dei confronti nell'ambito di ogni passata e quello dell'inversione.

Per risolvere il problema dell'evoluzione dei confronti, utilizziamo proprio il segnaposto naturale dei vettori, cioe' l'indice della posizione che, nell'ambito del vettore, identifica l'elemento che stiamo trattando.

In termini simbolici, se chiamiamo E la posizione generica, V(E) significa la variabile che occupa la posizione E e V(E+1) quella contigua di posto E+1. Inserendo questi elementi a indice variabile in un ciclo FOR...NEXT, all'evolvere dell'indice, evolve anche la coppia V(E) e V(E+1).

Per l'inversione, il meccanismo e' quello indicato in figura 9.10 e si avvale di un deposito ausiliario di nome y. (Al momento non utilizziamo ancora l'istruzione **SWAP** che inverte automaticamente i contenuti).

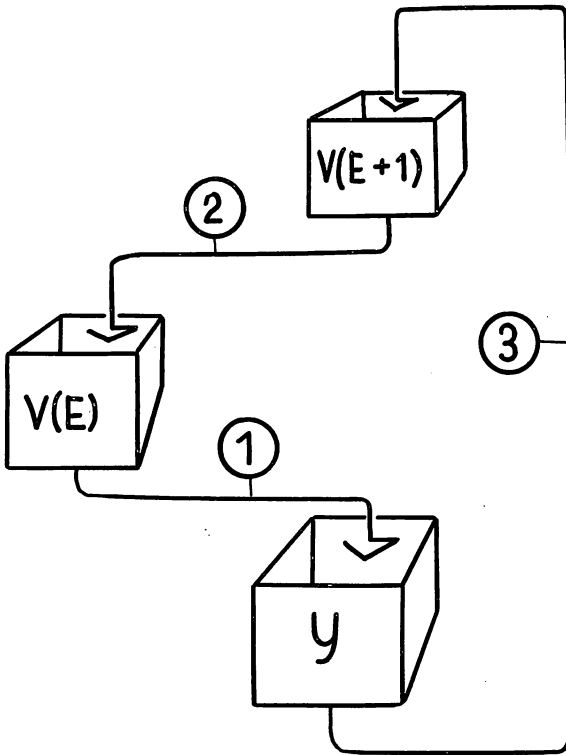



Figura 9.10

Siamo ora in grado di scrivere il programma, denominato **NORDOT**, che produce l'**ordinamento di N numeri** secondo l'algoritmo appena descritto, ma che non contiene ancora le migliorie accennate. In tale programma, qui di seguito riportato, la dimensione massima prevista per il vettore V e' di 100 elementi (linea 20). Come potete notare, nel corpo di buona parte delle linee di programma sono stati inseriti i commenti per illustrare la funzione che esse attivano.

```
10 'nordot: ordinamento non ottimizzato di N numeri
15 CLS:KEY OFF
20 DIM V(100)
30 INPUT "n. elementi=",N    'definisce il numero degli elementi
40 FOR E=1 TO N
50 INPUT ;           V(E)
55 NEXT
60 '*******"CICLO DELLE PASSATE"*******
70 FOR P=1 TO N-1  'ciclo interno dei confronti nell'ambito di ogni passata.
75                 'Caso di un un numero costante di confronti per passata.
80 FOR E=1 TO N-1
90 IF V(E) <= V(E+1) THEN 140
100 'inversione E in E+1 e viceversa
110 Y=V(E)     'salviamo il contenuto di v(E) in un deposito ausiliario Y
120 V(E)=V(E+1) 'il contenuto di E+1 va in V(E)
130 V(E+1)=Y    'il contenuto di Y va in V(E+1)
140 NEXT E
150 NEXT P
155 PRINT
160 '*******PROGRAMMA DI STAMPA*******
170 FOR E=1 TO N
180 PRINT V(E);  'il punto e virgola <;> fa si' che i numeri vengano stampati
185              'vicini e non su righe diverse, dedicate a ciascuno di essi.
190 NEXT E
200 END     'rimette a zero il contenuto di tutti i depositi
```

Con le linee 30, 40 e 50 abbiamo riprodotto una parte del programma lislo1 per il caricamento degli elementi della lista

Dalla linea 70 alla 150 corre il ciclo iterativo piu' esterno che gestisce le passate, mentre nel suo interno, dalla 80 alla 140, il corpo centrale costituito dalle istruzioni che confrontano 2 a 2 gli elementi, invertendoli con le istruzioni 110,120, 130. ogni volta che l'elemento di posto E e' maggiore di quello di posto E+1.

Al termine di entrambi i cicli FOR...NEXT si estrae il contenuto della lista con gli elementi ordinati.Tale visualizzazione avviene, con un ulteriore ciclo, alle istruzioni 170,180,190. La stampa dei risultati avviene in una riga diversa da quella di introduzione dei dati, a carico della PRINT alla linea 155.

Cerchiamo ora di mettere in pratica quelle migliorie a cui prima accennavamo. La prima riguarda la linea 80, nella quale porremo come limite alla variabile E non piu' un valore costante N-1, ma N-P. In tal modo per N=5, alla prima passata si faranno N-1=4 confronti; alla seconda, N-2=3 confronti; alla terza, N-3=2 confronti; alla quarta, infine, N-4=1 confronti.

Valutiamo il vantaggio ottenuto in termini di numero di confronti risparmiati. Nel programma di partenza, riferendoci sempre a N=5, il numero di confronti C1 era stato pari a 16 (4x4); col programma corretto invece scende a

$$C1' = N - 1 + N - 2 + N - 3 + N - 4 = (N - 1)\, N - (N/2)\,(N-1) =$$

$$= (N - 1)\, N/2 = 4\,(5/2) = 10$$

Per N = 100

C1 = 99 x 99 = 9801
C1'= 99 x 100 - 50 x 49 = 4950

La tabella di figura 9.11 mostra i valori di C1 e C1' per 4 valori di N da cui si vede che il numero di confronti cresce col quadrato degli elementi da ordinare.

N	5	11	21	101
C1 =(N- 1)^2	16	100	400	10000
C1'=(N-1)N/2	10	55	210	5050
C1-C1'	6	45	190	4950

Figura 9.11

Come abbiamo gia' spiegato, la seconda miglioria da apportare al programma NORDOT, e' di prevedere una scorciatoia che lo instradi rapidamente verso la fine, appena avverte che un'intera passata non ha prodotto alcuna inversione.
In altre parole, e' sufficiente inserire prima del riciclo dei confronti (140 NEXT E) un'istruzione che lasci la traccia tutte le volte che il gruppo delle istruzioni alla linee 120,130,140 viene eseguito.
Tale traccia e' un po' come un nodo al fazzoletto: puo' essere costituita assegnando un valore qualsiasi, diverso da zero, a una variabile T, ad esempio inserendo alla linea 135 un'istruzione di assegnazione T=1.
Se inoltre rimettiamo a zero T prima del ciclo delle passate, per esempio alla linea 75, avremo costruito il meccanismo che serve allo scopo.
Infatti, se dopo la serie di tutti i confronti di una passata la variabile T varra' ancora zero, significa che il programma non ha mai eseguito la linea 135, che serve appunto a assegnarle il valore 1. Cio' a causa della condizione 90 che, facendo saltare le istruzioni di inversione, fa saltare anche la 135.
La condizione posta alla linea 145, dopo una passata, verifica se T e' ancora a zero: in tal caso (assenza di nodo al fazzoletto) il programma, abbandonando le passate residue, esce per la scorciatoia saltando alla linea 160 dove trova le istruzioni per la stampa dei risultati.
Il nuovo programma, denominato **ORDOT**, cosi' ottenuto, con le modifiche apportate alla prima versione NORDOT, e' quindi il seguente:

```
10 'ordot:--------------------------------ordinamento  ottimizzato di N numeri
15 CLS:KEY OFF
20 DIM V(100)
30 INPUT "n. elementi=",N    'definisce il numero degli elementi
40 FOR E=1 TO N
50 INPUT ;             V(E)
55 NEXT
60 '*******"CICLO DELLE PASSATE"*********************************************
70 FOR P=1 TO N-1
75 T=0
80 FOR E=1 TO N-P  '..............................ciclo interno dei confronti
90 IF V(E) <= V(E+1) THEN 140
100 'inversione E in E+1 e viceversa
110 Y=V(E)     'salviamo il contenuto di v(E) in un deposito ausiliario Y
120 V(E)=V(E+1) 'il contenuto di E+1 va in V(E)
130 V(E+1)=Y    'il contenuto di Y va in V(E+1)
135 T=1
140 NEXT E         '..........................................................
145 IF T=0 THEN 160
150 NEXT P
160 PRINT'_______PROGRAMMA DI STAMPA________________________________________
170 FOR E=1 TO N
180 PRINT V(E);  'il punto e virgola <;> fa si' che i numeri vengano stampati
185              'vicini e non su righe diverse, dedicate a ciascuno di essi.
190 NEXT E
200 END                     'rimette a zero il contenuto di tutti i depositi
```

Abbiamo appena migliorato un programma e subito altre idee cominciano a farsi strada: una di queste e' legata all'efficienza del programma ORDOT.

Infatti, tra i principali fattori che influenzano il tempo di ordinamento ve n'e' uno che puo' essere facilmente controllato. Si tratta dei trasferimenti dei valori delle variabili, nell'ambito del meccanismo di inversione di ordot. Il tempo di esecuzione di questi trasferimenti dipende dalla lunghezza dell'elemento da trasferire: se solo riuscissimo a evitarli avremmo drasticamente ridotto il tempo di ordinamento.

A questo punto l'idea e' matura.

Bastera' crearsi una sorta di controfigura dell'elemento con la quale l'elemento reale condivida solo la posizione che occupa nella lista. Questa corrispondenza va quindi posta al momento del caricamento iniziale dei dati. E' cio' che viene rappresentato nella tabella di figura 9.12.

Come si puo' notare la **variabile indice** F (freccia) e' a tutti gli effetti un **puntatore** della posizione e quindi assume proprio il valore del suo stesso indice. Il termine generale F(E) = E e' cio' che mettiamo nel corpo della stessa FOR...NEXT insieme alla INPUT di chiamata dei dati.

	V(1)	V(2)	V(3)	V(4)	V(5)
V	7	15	9	6	4
F	F(1)	F(2)	F(3)	F(4)	F(5)
E	1	2	3	4	5

controfigura di V

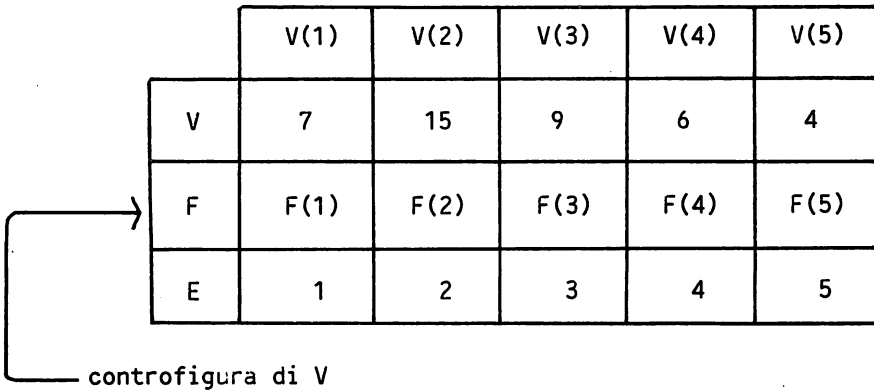



Figura 9.12

In tal modo, sempre con riferimento alla suddetta tabella, F(3) non rappresenta solo una variabile che contiene il suo indice, ma anche il vettore V(3) che contiene il valore 9, in quanto V(3) e' stato inizialmente associato a F(3).

Lo scopo di tutto cio' dovrebbe ora risultare chiaro, in quanto e' legato al ragionamento iniziale sull'utilita' di risparmiare al massimo i trasferimenti interni di dati e quindi all'obiettivo di ridurre il tempo di ordinamento.

Ovviamente, nelle operazioni di confronto gli elementi coinvolti non possono che essere quelli reali, in quanto riguardano i contenuti effettivi delle variabili in gioco; mentre ogni freccia, per definizione, contiene solo il numero d'ordine che occupava l'elemento al momento del suo caricamento iniziale.

Siamo cosi' giunti alla formalizzazione di un altro meccanismo che possiamo introdurre nel programma ORDOT. Questo, con le opportune modifiche e la rinumerazione delle linee, si trasforma nel nuovo programma **ordind** (**ord**inamento **ind**iretto ottimizzato di N numeri), che riportiamo qui di seguito.

```
10 'ordind:ordinamento indiretto ottimizz. di N numeri
15 CLS:KEY OFF
20 DIM V(100),F(100)
30 INPUT "N=",N
40 FOR E=1 TO N
50 INPUT; V(E)
60 F(E)=E
70 NEXT E
80 IF N=1 THEN 230
90 'ciclo delle passate .............................
100 FOR P=1 TO    N-1
110 T=0
```

```
120 'ciclo interno dei confronti < < < < < < < < < < <
130 FOR E=1 TO N-P
140 IF V(F(E)) <= V(F(E+1)) THEN 200
150 'inversione -------> <---------
160 Q=F(E)
170 F(E)=F(E+1)
180 F(E+1)=Q
190 T=1
200 NEXT E                      '> > > > > > > > > > >
210 IF T=0 THEN 230
220 NEXT P                 '..............................
230 '____________________programma di stampa___________
240 PRINT
250 FOR E=1 TO N
260 PRINT V(F(E));
270 NEXT E
280 END
Ok
RUN
N=8
? 5? 8? 76? 2? 57? 11? 1? 24
 1  2  5  8  11  24  57  76

Ok
RUN
N=5
? 12? 88? 54? 2? 27
 2  12  27  54  88

Ok
RUN
N=5
? 777? 13? 1956342943299? 12231? 25
 13  25  777  12231  1.956343E+12
```

Come potete vedere alla linea 20, anche il vettore freccia F e' stato dimensionato a 100 elementi come il vettore V; entrambi alle linee 50 e 60 evolvono nell'ambito di N cicli di caricamento, dove N e' il numero degli elementi generici del vettore V, il cui indirizzamento e' fatto con i contenuti del vettore F.

Dopo il listato di ordind abbiamo riportato alcune passate di prova; anche con introduzioni errate, per mettere in evidenza che gli elementi introdotti devono essere compatibili con la natura del vettore V di appartenenza.

Il meccanismo di indirizzamento indiretto viene illustrato nelle figure 9.13 e 9.14, con le quali, insieme all'evoluzione dei vettori V e F nei diversi stadi interni del processo, i due metodi descritti vengono tra loro contrapposti.

Si noti che facendo girare il programma con numeri come quelli della tabella di figura 9.13 non apparirebbe chiaramente il guadagno netto realizzato, perche' gli elementi reali del vettore V considerati nell'esempio sono numeri di una o due cifre; ma se fossero stringhe di 255 caratteri, il peso sarebbe assai diverso.

V(1)	V(2)	V(3)	V(4)	V(5)
7	15	9	6	4

P	E	F(1)	F(2)	V(3)	V(4)	V(5)
P	E	1	2	3	4	5
1	1 - 2	1	3	2	4	5
	3	1	3	4	2	5
	4	1	3	4	5	2
2	1 - 2	1	4	3	5	2
	3	1	4	5	3	2
3	1	4	1	5	3	2
	2	4	5	1	3	2
4	1	5	4	1	3	2
	V(F(E))	4	6	7	9	15

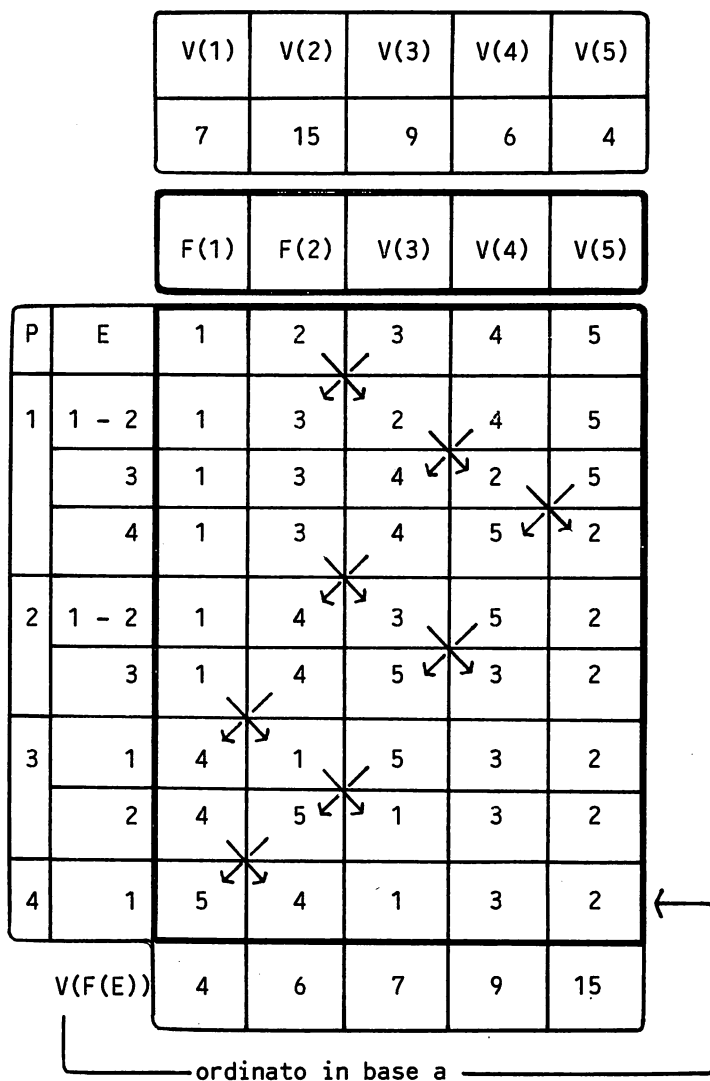



Figura 9.13

Nella tabella di figura 9.14, invece, si puo' seguire il succedersi delle disposizioni degli elementi secondo la procedura del programma NORDOT.

P	E	V(1)	V(2)	V(3)	V(4)	V(5)	
		7	15	9	6	4	disposizione iniziale
1	1	7	15	9	6	4	
	2	7	9	15	6	4	
	3	7	9	6	15	4	
	4	7	9	6	4	15	
2	1	7	9	6	4	15	
	2	7	6	9	4	15	
	3	7	6	4	9	15	
	4	7	6	4	9	15	
3	1	6	7	4	9	15	
	2	6	4	7	9	15	
	3	6	4	7	9	15	
	4	6	4	7	9	15	
4	1	4	6	7	9	15	risultato finale
	2	4	6	7	9	15	
	3	4	6	7	9	15	
	4	4	6	7	9	15	

Figura 9.14

Proviamo ora a definire diversamente il vettore V trasformandolo in vettore stringa: e' sufficente porre alla linea 15 l'istruzione DEFSTR V (DEFinisco STRinga V). L'effetto di tale istruzione e' che ora il il programma ordind, tutte le volte che elabora il vettore V, lo considera non piu' costituito da variabili numeriche in semplice precisione, ma variabili stringa. Abbiamo in tal modo evitato di introdurre modifiche in tutti i punti del programma in cui compare V, che avremmo dovuto trasformare in V$.

Ora che con ordind trattiamo, stringhe bisogna fare attenzione che in tal modo i confronti tra i numeri avvengono non piu' sul loro valore algebrico, ma sulla loro rappresentazione binaria in memoria secondo la codificazione ASCII.

Vediamo alcuni esempi di **ordinamento** eseguito **su stringhe** con la nuova versione del programma che viene ora chiamato **ordind1.**

```
10 'ordind1:ordinamento indiretto ottimizz. di N stringhe
15 CLS:KEY OFF:DEFSTR V
20 DIM V(100),F(100)
30 INPUT "N=",N
40 FOR E=1 TO N
50 LINE INPUT; "|",V(E)
60 F(E)=E
70 NEXT E
80 IF N=1 THEN 230
90 'ciclo delle passate ..............................
100 FOR P=1 TO    N-1
110 T=0
120 'ciclo interno dei confronti < < < < < < < < < < < <
130 FOR E=1 TO N-P
140 IF V(F(E)) <= V(F(E+1)) THEN 200
150 'inversione -------> <----------
160 Q=F(E)
170 F(E)=F(E+1)
180 F(E+1)=Q
190 T=1
200 NEXT E                      '> > > > > > > > > > > > >
210 IF T=0 THEN 230
220 NEXT P                      '..............................
230 '____________________programma di stampa____________
240 PRINT
250 FOR E=1 TO N
260 PRINT "|"V(F(E));
270 NEXT E
280 END
RUN
N=8
|I|cipressi|che|a|Bolgheri|alti|e|schietti
|Bolgheri|I|a|alti|che|cipressi|e|schietti

Ok
RUN
N=27
|q|w|e|r|t|y|u|i|o|p|a|s|d|f|g|h|j|k|l|z|x|c|v|b|n|m|
||a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

Ok
RUN
N=10
|1230a|123b|123c|123d|123e|123f|0123g|0.123h|.123i|.123j
|.123i|.123j|0.123h|0123g|1230a|123b|123c|123d|123e|123f

Ok
```

Si noti, nell'ultima passata, un esempio di successione di numeri (ogni numero, per comodita', porta in coda una lettera alfabetica che va d'accordo con l'ordine d'introduzione), alcuni dei quali, pur essendo equivalenti non sono considerati tali, per la logica dei confronti su stringhe che, come prima dicevamo in merito alla convenzione sui numeri, da' un peso diverso ai separatori tipo spazio, punto decimale e zero.

Vediamo ora una variante di ORDOT che e' costituita dal programma **ordolen** nel quale sono state apportate le seguenti modifiche.

- **Controllo della lunghezza della riga stampata** per evitare che, in assenza di ritorno a capo, le righe vengano sovrapposte. Il meccanismo e' un po' elaborato e contiene l'istruzione LEN (alla linea 65) per leggere la lunghezza della stringa Q ottenuta per concatenazione di quella attuale con la somma parziale di quella precedente (dopo l'ultimo ritorno a capo), man mano che vengono introdotte da tastiera. Quando tale lunghezza supera il valore che abbiamo assegnato alla variabile RIGA nella input di linea 35, allora (THEN) l'istruzione PRINT fa proseguire la visualizzazione sulla riga successiva. Segue la rimessa a zero della somma parziale Q$ con il caricamento della stringa nulla. Analogo meccanismo funziona per la stampa dei risultati.

- **Calcolo del tempo di ordinamento.** L'orologio di macchina viene rimesso a zero con la TIME$ alla linea 80, immediatamente prima che scatti il processo di ordinamento. Alla linea 260, terminato l'ordinamento, il programma torna a leggere l'orologio il cui contenuto viene stampato prima dell'emissione dei risultati.

Riportiamo qui di seguito il programma **ordolen** con alcuni dati di prova.

```
LOAD"ordolen
Ok
1LIST
10 'ordolen  :            ordinamento ottimizzato di N stringhe
12 CLS:KEY OFF'           con controllo della riga di stampa.
15 DEFSTR V,Y
20 DIM V(100),S$(100),SOR$(100)
30 INPUT; "n.elementi=",N  'definisce il numero degli elementi
35 INPUT "   lunghezza riga -> ",RIGA
40 FOR E=1 TO N
50 PRINT "...("E")=";
60 INPUT; "",V(E)
63 S$(E)=V(E)+Q$          'A ogni ciclo S$ accumula il nuovo ele-
64 Q$=S$(E)               'mento che viene salvato in Q$.
65 IF LEN (Q$) > RIGA  THEN PRINT :Q$=""  'Quando la somma par-
70 NEXT E                                 'ziale Q$ supera il
80 TIME$="00:00:00"                       'valore assegnato alla
90 '****ciclo delle passate****           'lunghezza della RIGA
100 FOR P=1 TO N-1                        'il cursore va a capo.
110 T=0
```

```
120 FOR E=1 TO N-P
130 IF V(E) <= V(E+1) THEN 190
140 'inversione E in E+1
150 Y=V(E)
160 V(E)=V(E+1)
170 V(E+1)=Y
180 T=1
190 NEXT E
200 IF T=0 THEN 230
220 NEXT P
230 '*****programma di stampa*****
240 PRINT
250 PRINT "tempo di ordin.=";
260 PRINT TIME$
270 PRINT
280 FOR E=1 TO N
290 PRINT V(E);
293 SOR$(E)=V(E)+QOR$                              '................
294 QOR$=SOR$(E)                                   '     i.c.s.
295 IF LEN (QOR$) > RIGA THEN PRINT: QOR$=""       '................
300 NEXT E
310 END
Ok_
RUN
n.elementi=10   lunghezza riga -> 8
...( 1 )=nel...( 2 )=mezzo...( 3 )=del
...( 4 )=cammin...( 5 )=di...( 6 )=nostra
...( 7 )=vita...( 8 )=mi...( 9 )=ritrovai
...( 10 )=per
tempo di ordin.=00:00:00

cammindel
dimezzomi
nelnostra
perritrovai
vita

Ok_
RUN
n.elementi=10   lunghezza riga -> 12
...( 1 )=nel...( 2 )=mezzo...( 3 )=del...( 4 )=cammin
...( 5 )=di...( 6 )=nostra...( 7 )=vita...( 8 )=mi
...( 9 )=ritrovai...( 10 )=per
tempo di ordin.=00:00:00

cammindeldimezzo
minelnostraper
ritrovaivita

Ok_
```

Per terminare il trattamento delle liste vediamo il seguente programma **anas2**. Con esso si puo' introdurre una lista di date dimensioni e, dopo averla ordinata, e' possibile registrarla con un dato nome.

Quando si voglia rileggere una lista da dischetto, basta richiamarne il nome per riaverla in memoria. Quando la lista registrata su dischetto non fosse gia' ordinata, la si puo' ordinare dopo il suo trasferimento in memoria.

Altra possibilita' e' quella di introdurre nuovi elementi in una lista gia' registrata su dischetto, senza dover ribattere gli elementi gia' presenti. Per riordinare la nuova lista dopo il caricamento dell'appendice, si richiama tutta la lista, la si riordina in memoria e poi la si rinvia su dischetto con lo stesso nome che aveva la lista prima dell'aggiornamento.

```
10 'anas2:Caricamento di una lista di stringhe e successiva registra-
20 '......zione su file seq.(resvet). (N)=nuova; (A)=aggior.; (L)=lett.
25 CLS:KEY OFF:F1$="a:":INPUT "nome del file = ",F2$:F$=F1$+F2$
30 M=ASC(F2$):IF (M<65 OR M>90) AND (M<96 OR M>122) GOTO 25
35 '
36 KEY OFF:INPUT "Nuova Registr.(O); Aggior.(A); Lett.(I) -->  ",G$
47 IF G$="I" THEN N=8000:B=0:GOTO 60
48 IF G$<>"O" AND G$<>"I" AND G$<>"A" THEN 35
50 INPUT "n.elem.=",N
60 DIM V$(N)
65 IF G$="I" THEN 180
90 FOR E=1 TO N
100 INPUT; V$(E)
120 NEXT E
121 PRINT:PRINT"Un tasto per continuare":W$=INPUT$(1)
122 PRINT:IF G$="I" THEN 180
125 D$=","
130 OPEN G$,1,F$'.............REGISTR./AGGIORN.
150 FOR E=1 TO N
160 PRINT# 1,V$(E);D$
170 NEXT:CLOSE:ERASE V$:GOTO 36
180 OPEN G$,1,F$ '.....................LETTURA
190 FOR E=1 TO N
192 B=B+1
195 IF EOF(1) THEN CLOSE:GOTO 213
200 INPUT# 1,V$(E)
202 'V1$(E)=V$(E)
205 PRINT V$(E);" ";
210 NEXT:CLOSE
213 PRINT:NB=(B-1)/2:PRINT NB
215 PRINT:ERASE V$':IF G$="I" THEN 36
219 'IF W$="S" THEN 250
255 DIM V1$(B),V$(B)
280 OPEN G$,1,F$ '...................RILETTURA
290 FOR E=1 TO N
292 'B=B+1
```

```
295 IF EOF(1) THEN CLOSE 1:GOTO 318
300 INPUT# 1,V$(E)
302 V1$(E)=V$(E)
305 'PRINT V$(E);" ";
310 NEXT:CLOSE
318 PRINT:PRINT "(S) per ordinare la lista....";
319 PRINT"Un tasto per altre scelte":W$=INPUT $(1)
320 IF W$<>"S" THEN ERASE V$,V1$:GOTO 36
500 'ordind1:ordinamento indiretto ottimizz. di N stringhe
510 PRINT "N="N;"B="B;"NB="NB
520 DIM F(B)
530 N=B'
540 FOR E=1 TO N
560 F(E)=E
570 NEXT E
580 IF N=1 THEN 730
590 'ciclo delle passate ...............................
600 FOR P=1 TO   N-1
610 T=0
620 'ciclo interno dei confronti < < < < < < < < < < < <
630 FOR E=1 TO N-P
640 IF V1$(F(E)) <= V1$(F(E+1)) THEN 700
650 'inversione -------> <----------
660 'Q=F(E)
670 SWAP F(E),F(E+1)
680 'F(E+1)=Q
690 T=1
700 NEXT E                 '> > > > > > > > > > > >
710 IF T=0 THEN 730
715 SOUND 2500,.4
720 NEXT P                 '...............................
730 '___________________programma di stampa____________
740 PRINT
750 FOR E=1 TO N
760 PRINT V1$(F(E));" ";
770 NEXT E
772 PRINT:PRINT:PRINT"(R)=registra ordinato"
773 W$=INPUT $(1):IF W$="R" THEN 800
775 ERASE V$,V1$,F:PRINT:GOTO 36
780 D$=","
800 OPEN"O",1,F$
810 FOR E=1 TO N
820 PRINT# 1,V1$(F(E));D$
830 NEXT:CLOSE:ERASE V$,V1$,F:GOTO 36
Ok_
```

Qui di seguito alcune passate del programma anas2.

```
RUN
nome del file = tast
Nuova Registr.(O); Aggior.(A); Lett.(I) -->  O
n.elem.=8
? q? w? e? r? t? y? u? i
Un tasto per continuare

Nuova Registr.(O); Aggior.(A); Lett.(I) -->  I
q  w  e  r  t  y  u  i
 8


(S) per ordinare la lista....Un tasto per altre scelte
N= 8000 B= 17 NB= 8

         e i q r t u w y

(R)=registra ordinato
Nuova Registr.(O); Aggior.(A); Lett.(I) -->  I
                 e  i  q  r  t  u  w  y
 17


(S) per ordinare la lista....Un tasto per altre scelte
Nuova Registr.(O); Aggior.(A); Lett.(I) -->  A
n.elem.=aa3
? dd? zzz? aaaa
Un tasto per continuare

Nuova Registr.(O); Aggior.(A); Lett.(I) -->  I
                 e  i  q  r  t  u  w  y  dd  zzz  aaaa
 20


(S) per ordinare la lista....Un tasto per altre scelte
N= 8000 B= 41 NB= 20

                              aaaa dd e i q r t u w y zzz

(R)=registra ordinato
Nuova Registr.(O); Aggior.(A); Lett.(I) -->  I
                              aaaa  dd  e  i  q  r
  t  u  w  y  zzz
 41


(S) per ordinare la lista....Un tasto per altre scelte
Nuova Registr.(O); Aggior.(A); Lett.(I) -->  I
                              aaaa  dd  e  i  q  r
  t  u  w  y  zzz
 41


(S) per ordinare la lista....Un tasto per altre scelte
```

Gli ordinamenti fin qui trattati hanno operato su liste di elementi o vettori. Vogliamo ora fabbricare un programma che ordini gli elementi di una tabella con opportuni criteri. Supponiamo di aver caricato in memoria una tabella di 5 righe x 4 colonne, come quella riportata in figura 9.15, corrispondente alla catalogazione di brani musicali su cassette.

In tale tabella le righe corrispondono ai brani musicali, mentre le divisioni create con le colonne descrivono particolari aspetti o attributi di ogni brano musicale. Tra questi attributi possiamo scegliere l'autore del brano, gli estremi dell'opera, il nome dell'esecutore e l'ubicazione del brano corrispondente alla codificazione adottata per etichettare la cassetta. Per semplicita', nella tabella abbiamo abbreviato questi attributi al minimo essenziale per limitare l'introduzione dei dati, ma il campo assegnato puo' essere molto piu' esteso (fino a 255 caratteri).

UBI	AUT	SOG	ESE
052	bet	not	cam
032	bac	toc	mic
014	sch	op2	cia
048	rac	s07	mag
004	cho	bal	pol

Figura 9.15

Per caricare in memoria questa tabella usiamo il programma tablo che gia' conosciamo. Pertanto nell'istruzione DIM dovremo definire entrambe le dimensioni della tabella. La dimensione del vettore F segnaposto sara' uguale a quella delle righe, perche' l'**ordinamento dovra'** essere eseguito **colonna per colonna.**

Per lo stesso motivo la creazione di F avviene parallelamente al caricamento delle righe della tabella. In figura 9.16 abbiamo rappresentato il concatenamento tra tabella e freccia segnaposto.

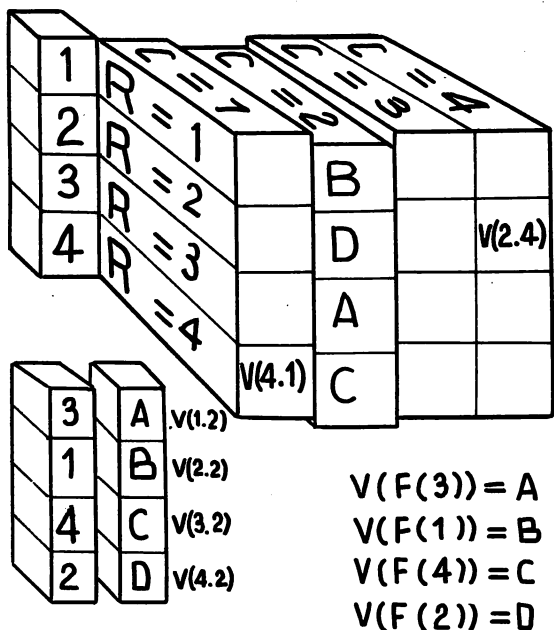



Figura 9.16

Il seguente programma, denominato **ositab** realizza l'**ordinamento indiretto di stringhe in una tabella** di 100 x 100.
Alla linea 170 avviene la selezione delle colonne, che verra' presa per riferimento dalla procedura dei confronti. Come nei precedenti programmi l'ordinamento e' indiretto e alla fine, secondo l'ordine dettato dal vettore freccia, vengono estratti gli elementi della tabella che **trascinano con se' gli elementi appartenenti alla stessa riga.**

```
10 'ositab:.....ordinamento indiretto di stringhe in Tab.(R X C)
20 CLS:KEY OFF'......inizializzazione e dimensionamento depositi
30 INPUT;"n.righe tab.=",RT
40 INPUT; "  n.colonne tab.=",CT
50 DIM V$(100,100)         'Da qui si comprende che l'ordinamento
60                         'verra' fatto colonna per colonna: in-
70                         'fatti creiamo una freccia F che scan-
71 PRINT                       'disca le righe a parita' di col.
72 '
73 'caricamento della tabella --------------------------------
74 '
80 FOR R=1 TO RT
91 '
100 FOR C=1 TO CT ' (inizia il ciclo interno trattam. col.)
110 PRINT "...V("R","C")=";
120 INPUT; "",V$(R,C)
```

```
121 PRINT "   ";
130 NEXT C          '(fine del ciclo interno di trattam. col.)
131 PRINT
132 '
140 NEXT R
155 DIM F(100):PRINT
156 '
157 FOR R=1 TO RT                  'Creazione freccia segnaposto
158 F(R)=R
159 NEXT R
160 'ordinamento colonna per colonna
170 INPUT "ordinamento per C=",C  'Assegnando un valore a C si
171                               'assume per riferimento quella
172                               'colonna e si mettono le righe
173                               'nello stesso ordine in cui
174                               'verra' ordinata la colonna
175                               'prescelta.
181 '
182 'Inizio tempo di ordinamento
185 TIME$="00:00:00
190 IF RT=1 THEN 330
195 '
196 'ciclo delle passate ---------------------------------------
200 FOR P=1 TO RT-1
205 '
210 T=0
211 '
220 'ciclo interno dei confronti
230 FOR R=1 TO RT-P
240 IF V$(F(R),C)  <= V$(F(R+1),C) THEN 300
250 'elem. R> elem.R+1 : INVERSIONE ===> <===
260 Q=F(R)
270 F(R)=F(R+1)
280 F(R+1)=Q
290 T=1
300 NEXT R
301 '
310 IF T=0 THEN 330
320 NEXT P
325 '
330 '**************programma di stampa*************************
331 '
340 PRINT
345 PRINT "tempo di ordinamento per V=(R,"C")==>";
347 PRINT TIME$
348 PRINT
350 FOR R=1 TO RT
355 PRINT F(R)"-->";
360 FOR C=1 TO CT
```

```
370 PRINT V$(F(R),C);
371 PRINT" ";    'distanzia gli elementi cosi'--> xx yy zz
380 NEXT C
382 PRINT
390 NEXT R
392 ERASE F
395 GOTO 155
396 ERASE F,V:GOTO 25
400 END
```

L'estrazione dei risultati ha inizio alla linea 350 con un doppio ciclo iterativo, nel piu' interno dei quali giace l'**elemento generico** della tabella con l'indice di riga **pilotato dalla freccia F(R).** Si noti che al termine della stampa , alla linea 395, s'incontra una GOTO che rinvia alla 155: cio' rende possibile la selezione su una nuova colonna con il preventivo ricaricamento della freccia F(R). Per tale motivo questo caricamento non e' stato incorporato nel ciclo di introduzione dei dati per la creazione della tabella.

Nelle prove effettuate sono emerse le note anomalie sull'ordinamento di numeri in regime di stringhe.

Come gia' detto in precedenza, occorre prendere delle precauzioni quando vogliamo trattare numeri come variabili stringa. I risultati, infatti, possono venire alterati, perche' la macchina nel corpo di una stringa considera diversi lo spazio, il punto decimale e lo zero. Cosi', ad esempio,per confrontare due numeri come 21 e 110, in regime stringhe, bisogna introdurre 021 e 110, altrimenti 21 risultera' maggiore di 110.

Per poter usare correttamente i numeri con le consuete convenzioni, senza restrizioni nell'introduzione dei dati, ositab e' stato opportunamente modificato dando cosi' luogo al nuovo programma di nome **ortab.**

Parallelamente, con altre modifiche, e' stato messo a punto un altro programma di tipo generale, denominato **gentab,** che ha le seguenti caratteristiche.

1) Con opportune selezioni interne tratta numeri e stringhe, effettuando anche alcune protezioni sugli eventuali valori introdotti non ammessi. Di particolare rilievo, anche, l'autodimensionamento con la chiamata dei valori RT e CT che vengono assegnati nell'ambito della DIM.

2) E' stato previsto un meccanismo per riprendere l'intera procedura dall'inizio, mediante la selezione di un codice di colonna fittizio (C=99). In assenza di questo artificio occorreva impostare CTRL C seguito da un RUN. L'intercettazione di C=99 alla linea 174 rinvia alla linea 396 alla quale, dopo gli azzeramenti, si rinvia alla linea 25.

3) Alle linee 35 e 45 vengono filtrati eventuali valori nulli o negativi di RT e CT: la loro intercettazione fa ripetere l'in-

troduzione del dato.

4) Per ottenere particolare efficacia nella visualizzazione dei dati sono state accoppiate una Print e una Input (linee 110 e 115): in tal modo durante l'introduzione dei dati si segue fàcilmente l'evoluzione delle righe e delle colonne.

Entrambi i programmi ortab e gentab sono qui interamente riprodotti insieme ad alcuni dati di prova. Per ortab i dati di prova sono quelli di una matrice numerica 5 x 5; per gentab la prova riguarda una matrice rettangolare mista (alfanumerica) 9 x 6. Il contenuto di questa tabella riguarda un campione di brani musicali registrati su cassetta.

Tale tabella contiene:

col.1	col.2	col.3	col.4	col.5	col.6
AUTORE	ESECUTORE	OPERA	UBICAZIONE	DURATA	NOTE

Un esempio di introduzione dei dati di un brano musicale, corrispondente a una riga della tabella, e' il seguente:

```
- autore       (colonna 1)      Chopin
- esecutore    (   "    2)      Gulda
- opera        (   "    3)      24 preludi
- ubicazione   (   "    4)      34a030
- durata       (   "    5)      (in minuti)
- note         (   "    6)      eventuali
```

Poiche' la durata e l'ubicazione hanno campi numerici di selezione e' importante introdurle a lunghezza costante aggiungendo anche gli zeri non significativi alla sinistra della parte intera.

```
10 'ortab:ordinamento indiretto di NUMERI in Tab.(R X C)
20 'inizializzazione e dimensionamento depositi
25 CLS
30 INPUT;"n.righe tab.=",RT
40 INPUT; "  n.colonne tab.=",CT
50 DIM V(RT,CT),F(RT)    '  Da qui si comprende che l'ordinamento
60                       '  verra' fatto colonna per colonna: in-
70                       '  fatti creiamo una freccia F che scan-
71 PRINT                 '  disca le righe A PARITA' DI COLONNA.
72 '
73 'caricamento della tabella ---------------------------------
74 '
80 FOR R=1 TO RT
```

```
91 '
100 FOR C=1 TO CT ' (inizia il ciclo interno trattam. col.)
110 PRINT "...V("R","C")=";
120 INPUT; "",V(R,C)
121 PRINT "   ";
130 NEXT C          '(fine del ciclo interno di trattam. col.)
131 PRINT
132 '
140 NEXT R
155 PRINT
156 '
157 FOR R=1 TO RT                    'Creazione freccia segnaposto
158 F(R)=R
159 NEXT R
160 'ordinamento colonna per colonna
170 INPUT "ordinamento per C=",C  'Assegnando un valore a C si
171                               'assume per riferimento quella
172                               'colonna e si mettono le righe
173                               'nello stesso ordine in cui
174 IF C>=99 GOTO 396             'verra' ordinata la colonna
175 IF C>CT GOTO 170              'prescelta.
181 '
182 'Inizio tempo di ordinamento
185 TIME$="00:00:00
190 IF RT=1 THEN 330
195 '
196 'ciclo delle passate ----------------------------------------
200 FOR P=1 TO RT-1
205 '
210 T=0
211 '
220 'ciclo interno dei confronti
230 FOR R=1 TO RT-P
240 IF V(F(R),C)  <= V(F(R+1),C) THEN 300
250 'elem. R> elem.R+1 : INVERSIONE ===> <===
260 Q=F(R)
270 F(R)=F(R+1)
280 F(R+1)=Q
290 T=1
300 NEXT R
301 '
310 IF T=0 THEN 330
320 NEXT P
325 '
330 '**************programma di stampa**************************
331 '
340 PRINT
345 PRINT "tempo di ordinamento per V=(R,"C")==>";
347 PRINT TIME$
348 PRINT
350 FOR R=1 TO RT
355 PRINT F(R)"-->";
360 FOR C=1 TO CT
```

```
370 PRINT V(F(R),C);
371 PRINT" ";    'distanzia gli elementi cosi'--> xx yy zz
380 NEXT C
382 PRINT
390 NEXT R
392 ERASE F
395 GOTO 155
396 ERASE F,V:GOTO 25
400 END
```

```
RUN
n.righe tab.=4
```

```
          RUN
n.righe tab.=4  n.colonne tab.=4
...V( 1 , 1 )=12   ...V( 1 , 2 )=145   ...V( 1 , 3 )=124   ...V( 1 , 4 )=1277
...V( 2 , 1 )=456   ...V( 2 , 2 )=14.4   ...V( 2 , 3 )=44.04   ...V( 2 , 4 )=10

...V( 3 , 1 )=21.04   ...V( 3 , 2 )=214   ...V( 3 , 3 )=12.4   ...V( 3 , 4 )=9

...V( 4 , 1 )=14   ...V( 4 , 2 )=66.1   ...V( 4 , 3 )=514.1   ...V( 4 , 4 )=312


ordinamento per C=1

tempo di ordinamento per V=(R, 1 )==>00:00:00

 1 --> 12   145   124   1277
 4 --> 14   66.1   514.1   312
 3 --> 21.04   214   12.4   9
 2 --> 456   14.4   44.04   10

ordinamento per C=2

tempo di ordinamento per V=(R, 2 )==>00:00:00

 2 --> 456   14.4   44.04   10
 4 --> 14   66.1   514.1   312
 1 --> 12   145   124   1277
 3 --> 21.04   214   12.4   9

ordinamento per C=3

tempo di ordinamento per V=(R, 3 )==>00:00:00

 3 --> 21.04   214   12.4   9
 2 --> 456   14.4   44.04   10
 1 --> 12   145   124   1277
 4 --> 14   66.1   514.1   312

ordinamento per C=4

tempo di ordinamento per V=(R, 4 )==>00:00:00

 3 --> 21.04   214   12.4   9
 2 --> 456   14.4   44.04   10
 4 --> 14   66.1   514.1   312
 1 --> 12   145   124   1277

ordinamento per C=
Break in 170_

Ok_
```

Segue il programma **gentab.**

```
10 'gentab:ordinamento per colonna di una generica tabella (RxC)
20 'inizializzazione e dimensionamento depositi
25 CLS:KEY OFF
26 INPUT "PER INPUT NUMERICO PREMERE n e poi <CR> : ",M$ : PRINT
27 IF M$="n" GOTO 30
28 DEFSTR V
30 INPUT;"n.righe tab.=",RT
35 IF (RT=<0)  THEN LOCATE 3,1: GOTO 30
40 INPUT; "    n.colonne tab.=",CT
45 IF CT=<0 THEN LOCATE 3,16:GOTO 40
47 IF CT=99 THEN CT=0:LOCATE 3,18:GOTO 40
50 DIM V(RT,CT)'           Da qui si comprende che l'ordinamento
60                       '  verra' fatto colonna per colonna: in-
70                       '  fatti creiamo una freccia F che scan-
71 PRINT:PRINT           '  disca le righe A PARITA' DI COLONNA.
72 '
73 'caricamento della tabella ---------------------------------
74 '
80 FOR R=1 TO RT
91 '
100 FOR C=1 TO CT      ' (inizia il ciclo interno trattam. col.)
110 PRINT "...V("R","C")=";
120 INPUT; "",V(R,C)
121 PRINT "   ";
130 NEXT C             '(fine del ciclo interno di trattam. col.)
131 PRINT
132 '
140 NEXT R
155 DIM F(RT):PRINT
156 '
157 FOR R=1 TO RT                     'Creazione freccia segnaposto
158 F(R)=R
159 NEXT R
160 'ordinamento colonna per colonna
170 INPUT "ordinamento per C=",C  'Assegnando un valore a C si
171                               'assume per riferimento quella
172                               'colonna e si mettono le righe
173                               'nello stesso ordine in cui
174 IF C>=99 GOTO 396             'verra' ordinata la colonna
175 IF C>CT GOTO 170              'prescelta.
181 '
182 'Inizio tempo di ordinamento
185 TIME$="00:00:00
190 IF RT=1 THEN 330
195 '
196 'ciclo delle passate ---------------------------------------
200 FOR P=1 TO RT-1
205 '
210 T=0
211 '
```

```
220 'ciclo interno dei confronti
230 FOR R=1 TO RT-P
240 IF V(F(R),C)  <= V(F(R+1),C) THEN 300
250 'elem. R> elem.R+1 : INVERSIONE ===> <===
260 Q=F(R)
270 F(R)=F(R+1)
280 F(R+1)=Q
290 T=1
300 NEXT R
301 '
310 IF T=0 THEN 330
320 NEXT P
325 '
330 '*************programma di stampa*************************
331 '
340 PRINT
345 PRINT "tempo di ordinamento per V=(R,"C")==>";
347 PRINT TIME$
348 PRINT
350 FOR R=1 TO RT
355 PRINT F(R)"-->";
360 FOR C=1 TO CT
370 PRINT V(F(R),C);
371 PRINT "  ";     'distanzia gli elementi cosi'--> xx  yy  zz
380 NEXT C
381 PRINT "   ";
382 PRINT
390 NEXT R
392 ERASE F
395 GOTO 155
396 ERASE F,V:GOTO 25
400 END
```

```
RUN
PER INPUT NUMERICO PREMERE n e poi <CR> : 2

n.righe tab.=9    n.colonne tab.=6

...V( 1 , 1 )=hayd   ...V( 1 , 2 )=dora   ...V( 1 , 3 )=si49   ...V( 1 , 4 )=32a
019   ...V( 1 , 5 )=20   ...V( 1 , 6 )=
...V( 2 , 1 )=mahl   ...V( 2 , 2 )=klem   ...V( 2 , 3 )=si9   ...V( 2 , 4 )=60b0
00   ...V( 2 , 5 )=90   ...V( 2 , 6 )=mono
...V( 3 , 1 )=moza   ...V( 3 , 2 )=boeh   ...V( 3 , 3 )=s27   ...V( 3 , 4 )=15a0
00   ...V( 3 , 5 )=18   ...V( 3 , 6 )=chop.7
...V( 4 , 1 )=chop   ...V( 4 , 2 )=mich   ...V( 4 , 3 )=ma+g   ...V( 4 , 4 )=34b
052   ...V( 4 , 5 )=08   ...V( 4 , 6 )=0p.67
...V( 5 , 1 )=rave   ...V( 5 , 2 )=larr   ...V( 5 , 3 )=gasp   ...V( 5 , 4 )=43b
053   ...V( 5 , 5 )=20   ...V( 5 , 6 )=
...V( 6 , 1 )=hayd   ...V( 6 , 2 )=szel   ...V( 6 , 3 )=si-a   ...V( 6 , 4 )=18a
030   ...V( 6 , 5 )=21   ...V( 6 , 6 )=
...V( 7 , 1 )=chop   ...V( 7 , 2 )=guld   ...V( 7 , 3 )=pr24   ...V( 7 , 4 )=34a
030   ...V( 7 , 5 )=18   ...V( 7 , 6 )=cnt-b
...V( 8 , 1 )=moza   ...V( 8 , 2 )=mari   ...V( 8 , 3 )=ns+d   ...V( 8 , 4 )=15a
040   ...V( 8 , 5 )=25   ...V( 8 , 6 )=k286
...V( 9 , 1 )=chop   ...V( 9 , 2 )=ashk   ...V( 9 , 3 )=ba+f   ...V( 9 , 4 )=34b
150   ...V( 9 , 5 )=15   ...V( 9 , 6 )=op38/2
```

```
ordinamento per C=1

tempo di ordinamento per V=(R, 1 )==>00:00:00

 4 -->chop  mich  ma+g  34b052  08  Op.67
 7 -->chop  guld  pr24  34a030  18  cnt-b
 9 -->chop  ashk  ba+f  34b150  15  op38/2
 1 -->hayd  dora  si49  32a019  20
 6 -->hayd  szel  si-a  18a030  21
 2 -->mahl  klem  si9  60b000  90  mono
 3 -->moza  boeh  s27  15a000  18  chop.7
 8 -->moza  mari  ns+d  15a040  25  k286
 5 -->rave  larr  gasp  43b053  20

ordinamento per C=2

tempo di ordinamento per V=(R, 2 )==>00:00:00

 9 -->chop  ashk  ba+f  34b150  15  op38/2
 3 -->moza  boeh  s27  15a000  18  chop.7
 1 -->hayd  dora  si49  32a019  20
 7 -->chop  guld  pr24  34a030  18  cnt-b
 2 -->mahl  klem  si9  60b000  90  mono
 5 -->rave  larr  gasp  43b053  20
 8 -->moza  mari  ns+d  15a040  25  k286
 4 -->chop  mich  ma+g  34b052  08  Op.67
 6 -->hayd  szel  si-a  18a030  21

ordinamento per C= 3

tempo di ordinamento per V=(R, 3 )==>00:00:00

 9 -->chop  ashk  ba+f  34b150  15  op38/2
 5 -->rave  larr  gasp  43b053  20
 4 -->chop  mich  ma+g  34b052  08  Op.67
 8 -->moza  mari  ns+d  15a040  25  k286
 7 -->chop  guld  pr24  34a030  18  cnt-b
 3 -->moza  boeh  s27  15a000  18  chop.7
 6 -->hayd  szel  si-a  18a030  21
 1 -->hayd  dora  si49  32a019  20
 2 -->mahl  klem  si9  60b000  90  mono

ordinamento per C=4

tempo di ordinamento per V=(R, 4 )==>00:00:00

 3 -->moza  boeh  s27  15a000  18  chop.7
 8 -->moza  mari  ns+d  15a040  25  k286
 6 -->hayd  szel  si-a  18a030  21
 1 -->hayd  dora  si49  32a019  20
 7 -->chop  guld  pr24  34a030  18  cnt-b
 4 -->chop  mich  ma+g  34b052  08  Op.67
 9 -->chop  ashk  ba+f  34b150  15  op38/2
 5 -->rave  larr  gasp  43b053  20
 2 -->mahl  klem  si9  60b000  90  mono
```

```
ordinamento per C=2

tempo di ordinamento per V=(R, 2 )==>00:00:00

 9 -->chop  ashk  ba+f  34b150  15  op38/2
 3 -->moza  boeh  s27  15a000  18  chop.7
 1 -->hayd  dora  si49  32a019  20
 7 -->chop  guld  pr24  34a030  18  cnt-b
 2 -->mahl  klem  si9  60b000  90  mono
 5 -->rave  larr  gasp  43b053  20
 8 -->moza  mari  ns+d  15a040  25  k286
 4 -->chop  mich  ma+g  34b052  08  Op.67
 6 -->hayd  szel  si-a  18a030  21
```

```
ordinamento per C=5

tempo di ordinamento per V=(R, 5 )==>00:00:00

 4 -->chop  mich  ma+g  34b052  08  Op.67
 9 -->chop  ashk  ba+f  34b150  15  op38/2
 3 -->moza  boeh  s27  15a000  18  chop.7
 7 -->chop  guld  pr24  34a030  18  cnt-b
 1 -->hayd  dora  si49  32a019  20
 5 -->rave  larr  gasp  43b053  20
 6 -->hayd  szel  si-a  18a030  21
 8 -->moza  mari  ns+d  15a040  25  k286
 2 -->mahl  klem  si9  60b000  90  mono
```

```
ordinamento per C=6

tempo di ordinamento per V=(R, 6 )==>00:00:00

 1 -->hayd  dora  si49  32a019  20
 5 -->rave  larr  gasp  43b053  20
 6 -->hayd  szel  si-a  18a030  21
 4 -->chop  mich  ma+g  34b052  08  Op.67
 3 -->moza  boeh  s27  15a000  18  chop.7
 7 -->chop  guld  pr24  34a030  18  cnt-b
 8 -->moza  mari  ns+d  15a040  25  k286
 2 -->mahl  klem  si9  60b000  90  mono
 9 -->chop  ashk  ba+f  34b150  15  op38/2
```

Seguono ora due programmi di utilita', che costituiscono la sintesi di tutto quanto esposto in questo capitolo; si tratta dei programmi **resmor** e **resort**. Il primo serve a:

- caricare una tabella (R x C) di stringhe;

- ordinare le righe secondo una colonna desiderata;

- stampare le righe ordinate secondo la colonna prescelta;

- registrare la tabella ordinata in un file sequenziale.

Quando si vuole aggiornare il file registrato con resmor, serve, invece, il resort, con il quale, attraverso una scelta sul tipo di accesso al file, e' anche possibile eseguirne la sola lettura.

```
10 'resmor2:  registrazione su file sequenziale di tab.(R X C)
20 '                       ordinata secondo una colonna.
30 '--------------------inizializz. e dimensionamento depositi
40 CLS:KEY OFF
50 INPUT;"n.righe tab.=",RT
60 INPUT "  n.colonne tab.=",CT
70 DIM V$(RT,CT),F(RT) 'Da qui si comprende che l'ordinamento
80                     'verra' fatto colonna per colonna: in-
90                     'fatti creiamo una freccia F che scan-
100                    'disca le righe a parita' di col.
105 PRINT               "------------------------------------"
110 '
120 'caricamento della tabella -----------------------------
130 '
140 FOR R=1 TO RT
150 '
160 FOR C=1 TO CT ' (inizia il ciclo interno trattam. col.)
170 PRINT "...V("R","C")=";
180 LINE INPUT; "",V$(R,C)
185 IF V$(R,C)="" GOTO 180
190 PRINT "   ";
200 NEXT C          '(fine del ciclo interno di trattam. col.)
210 PRINT
220 '
230 NEXT R
240 PRINT
250 '
260 FOR R=1 TO RT                   'Creazione freccia segnaposto
270 F(R)=R
280 NEXT R
290 'ordinamento colonna per colonna
300 INPUT; "ordinamento per C=",C  'Assegnando un valore a C si
310                                 'assume per riferimento quella
320                                 'colonna e si mettono le righe
330                                 'nello stesso ordine in cui
340                                 'verra' ordinata la colonna
350 IF C=99 GOTO 790                'prescelta.
```

```
LIST 360-
360 IF C>CT GOTO 240
370 'Inizio tempo di ordinamento
380 TIME$="00:00:00"
390 IF RT=1 THEN 570
400 '
410 'ciclo delle passate -------------------------------------
415 TT=0
420 FOR P=1 TO RT-1
430 '
440 T=0
450 '
460 'ciclo interno dei confronti
470 FOR R=1 TO RT-P
480 IF V$(F(R),C)  <= V$(F(R+1),C) THEN 520
490 'elem. R> elem.R+1 : INVERSIONE ===> <===
500 SWAP F(R),F(R+1)
510 T=1
520 NEXT R
530 '
540 IF T=0 THEN 570
545 SOUND 1000,.5
546 TT=TT+1: PRINT TT; "<"TIME$">";
550 NEXT P
560 '
570 '****programma di stampa****
580 PRINT "                        "P"pass.      "TIME$ : PRINT
585 PRINT "PER STAMPARE PREMI UN TASTO QUALSIASI":Z$=INPUT$(1)
587 PRINT"------------------------------------------------------"
590 FOR R=1 TO RT
600 FOR C=1 TO CT
610 PRINT V$(F(R),C)"   ";
620 NEXT C
630 PRINT "   ";
640 PRINT
650 NEXT R
655 PRINT"------------------------------------------------------"
660 'programma di registr. su file sequenziale
670 PRINT "SE REG. INTRODURRE R (MAIUSCOLO!)"
675 CLOSE 1
680 A$=INPUT$(1)
690 IF A$="R" GOTO 700 ELSE 240
700 F1$="a:":INPUT "nome file =",F2$:F$=F1$+F2$
705 M=ASC(F2$):IF (M<65 OR M>90) AND (M<96 OR M>122) GOTO 700
710 OPEN "O",1,F$
712 PRINT# 1,RT:PRINT# 1,CT
715 D$=","
720 FOR R=1 TO RT: FOR C=1 TO CT
730 PRINT# 1, V$(F(R),C);D$
740 NEXT C
770 NEXT R: CLOSE
780 GOTO 240
Ok_
```

```
ordinamento per C=0                     1 pass.       00:00:00

PER STAMPARE PREMI UN TASTO QUALSIASI
--------------------------------------------------
Nel   mezzo   del   cammin   di   nostra   vita
mi   ritrovai   per   una   selva   oscura,   che'
la   diritta   via   era   smarrita.   Ah   quanto
a   dir   qual   era   e'   cosa   dura
esta   selva   selvaggia   e   aspra   e   forte
che   nel   pensier   rinova   la   paura!   Tant'e'
amara   che   poco   e'   piu'   morte;   ma
per   trattar   del   bene   ch'io   vi   trovai,
diro'   de   l'altre   cose   ch'io   v'ho   scorte
--------------------------------------------------
SE REG. INTRODURRE R (MAIUSCOLO!)

ordinamento per C=3 1 <00:00:00> 2 <00:00:00> 3 <00:00:00> 4 <00:00:00> 5
00:00:00> 6 <00:00:00>                        7 pass.       00:00:00

PER STAMPARE PREMI UN TASTO QUALSIASI
--------------------------------------------------
Nel   mezzo   del   cammin   di   nostra   vita
per   trattar   del   bene   ch'io   vi   trovai,
diro'   de   l'altre   cose   ch'io   v'ho   scorte
che   nel   pensier   rinova   la   paura!   Tant'e'
mi   ritrovai   per   una   selva   oscura,   che'
amara   che   poco   e'   piu'   morte;   ma
a   dir   qual   era   e'   cosa   dura
esta   selva   selvaggia   e   aspra   e   forte
la   diritta   via   era   smarrita.   Ah   quanto
--------------------------------------------------
SE REG. INTRODURRE R (MAIUSCOLO!)

ordinamento per C=5 1 <00:00:00> 2 <00:00:00> 3 <00:00:00> 4 <00:00:00> 5
00:00:00> 6 <00:00:00>                        7 pass.       00:00:00

PER STAMPARE PREMI UN TASTO QUALSIASI
--------------------------------------------------
esta   selva   selvaggia   e   aspra   e   forte
per   trattar   del   bene   ch'io   vi   trovai,
diro'   de   l'altre   cose   ch'io   v'ho   scorte
Nel   mezzo   del   cammin   di   nostra   vita
a   dir   qual   era   e'   cosa   dura
che   nel   pensier   rinova   la   paura!   Tant'e'
amara   che   poco   e'   piu'   morte;   ma
mi   ritrovai   per   una   selva   oscura,   che'
la   diritta   via   era   smarrita.   Ah   quanto
--------------------------------------------------
SE REG. INTRODURRE R (MAIUSCOLO!)

ordinamento per C=6 1 <00:00:00> 2 <00:00:00> 3 <00:00:00>
pass.       00:00:00
```

```
SE REG. INTRODURRE R (MAIUSCOLO!)

ordinamento per C=7 1 <00:00:00> 2 <00:00:00> 3 <00:00:00> 4 <00:00:00> 5 <
00:00:00>                    6 pass.     00:00:00

PER STAMPARE PREMI UN TASTO QUALSIASI
--------------------------------------------------
che   nel   pensier   rinova   la   paura!   Tant'e'
mi   ritrovai   per   una   selva   oscura,   che'
a   dir   qual   era   e'   cosa   dura
esta   selva   selvaggia   e   aspra   e   forte
amara   che   poco   e'   piu'   morte;   ma
la   diritta   via   era   smarrita.   Ah   quanto
diro'   de   l'altre   cose   ch'io   v'ho   scorte
per   trattar   del   bene   ch'io   vi   trovai,
Nel   mezzo   del   cammin   di   nostra   vita
--------------------------------------------------
SE REG. INTRODURRE R (MAIUSCOLO!)
nome file =Inferno
```

Il programma **resmor** e' chiaramente una composizione di altri programmi gia' visti; in particolare, le parti relative all'introduzione dei dati e la logica degli ordinamenti sono analoghe a quelle previste per gentab.

Alcune novita' riguardano il **meccanismo di inversione** che usa l'istruzione **SWAP**, mediante la quale si migliora di circa il 15 per cento il tempo di ordinamento. Quando si introducono tabelle di una certa dimensione, e quindi con tempi di ordinamento espressi in minuti, sono utili le **tracce sonora** e temporale, realizzate rispettivamente alle linee 545 e 546. Piu' precisamente, la 545 emette un suono di 1000 Hz di durata .1 sec, mentre la 546 visualizza il contenuto corrente della variabile TT, associandogli il valore attuale dell'orologio di macchina.

In tal modo, anche per attese di alcuni minuti non si ha il dubbio che la macchina si sia arrestata per qualche motivo; inoltre, verso la fine degli ordinamenti **la frequenza dei bip aumenta,** il che risveglia l'attenzione dell'operatore.

Per quanto riguarda la registrazione su file, si suppone che il volume coincida con quello fisico del dischetto inserito nel drive 1.

La linea 705 interroga la variabile M (equivalente all'entrata decimale, nella tabella ASCII, corrispondente al primo carattere del nome del file digitato) per verificare che tale carattere sia effettivamente alfabetico. Infatti il Basic non ammette nomi di variabili o di file che iniziano con altri caratteri. Ora, le frasi condizionali scritte alla 705 controllano appunto che M cada negli intervalli corrispondenti alle entrate numeriche delle lettere alfabetiche maiuscole e minuscole.

Si noti, inoltre, la concatenazione della stringa F2$, relativa al nome introdotto con quella F1$ che funge da prefisso: in tal modo, se tutto va bene, F$ (nome del file che compare nella OPEN alla 710) sara' proprio "a:F2$.

Il trasferimento su dischetto dei vari elementi della tabella ordinata in memoria, avviene mediante un doppio ciclo intorno alla linea 730. Rispetto al programma prores utilizzato come esempio di registrazione, resmor2 tratta, invece, una tabella: la logica della PRINT e del relativo separatore "," -definito come variabile D$ alla linea 715 e piazzato tra i vari elementi- e' da valutare con molta attenzione, perche' da questo artificio dipende il modo in cui avverra' la registrazione e la successiva rilettura.

Segue ora il programma **resort9**, che serve a leggere o ad aggiornare un file gia' registrato su dischetto. Nel programma sono aggregate parti di resmor2 e di anas2, opportunamente modificati.

```
5 'resort9 ---->     lettura di stringhe registrate in tabella
20 CLS:KEY OFF'      su file sequenziale.
30 CLOSE 1:INPUT;"nome testo-->",T$
40 OPEN "I",1,T$
1120 'caricamento della tabella ----------------------------
1130 INPUT#1,RT:INPUT #1,CT
1135 DIM V$(RT,CT),F(RT)
1140 FOR R=1 TO RT
1160 FOR C=1 TO CT ' (inizia il ciclo interno trattam. col.)
1205 LINE INPUT# 1,V$(R,C)
1210 NEXT C         '(fine del ciclo interno di trattam. col.)
1220 'PRINT
1230 '
1240 NEXT R:IF EOF(1) GOTO 1250
1250 PRINT
1260 '
1270 FOR R=1 TO RT                      'Creazione freccia segnaposto
1280 F(R)=R
1290 NEXT R
1300 'ordinamento colonna per colonna
1310 INPUT; "ordinamento per C=",C  'Assegnando un valore a C si
1320                                'assume per riferimento quella
1330                                'colonna e si mettono le righe
1340                                'nello stesso ordine in cui
1350                                'verra' ordinata la colonna
1360 IF C=99 GOTO 1840               'prescelta.
1370 IF C>CT GOTO 1250
1380 'Inizio tempo di ordinamento
1390 TIME$="00:00:00
1400 IF RT=1 THEN 1580
1410 '
1420 'ciclo delle passate ---------------------------------------
1430 FOR P=1 TO RT-1
1440 '
1450 T=0
1460 '
1470 'ciclo interno dei confronti
1480 FOR R=1 TO RT-P
1490 IF V$(F(R),C)  <= V$(F(R+1),C) THEN 1530
```

```
1500 'elem. R> elem.R+1 : INVERSIONE ===> <===
1510 SWAP F(R),F(R+1)
1520 T=1
1530 NEXT R
1540 '
1550 IF T=0 THEN 1580
1560 NEXT P
1570 '
1580 '****programma di stampa****
1590 PRINT "                          "P"pass.        "TIME$ : PRINT
1600 PRINT "PER STAMPARE PREMI UN TASTO QUALSIASI":Z$=INPUT$(1)
1610 PRINT"-----------------------------------------------------"
1612 FOR R=1 TO RT
1614 FOR C=1 TO CT
1616 'PRINT "...V("R","C")=";
1617 'res2
1619 C$=","
1621 P=0
1623 F=1
1625 S$=V$(F(R),C):V=INSTR(F,S$,C$)
1627 IF V=0 GOTO 1635
1629 P=P+1
1631 F=V+1
1633 GOTO 1625
1635 IF P=0 THEN 1650
1637 MID$(S$,F-1,1)=CHR$(0)
1639 PRINT S$" ";
1640 'PRINT V$(F(R),C)" ";
1650 NEXT C
1660 'PRINT "   ";
1670 PRINT
1680 NEXT R
1690 PRINT"-----------------------------------------------------"
1700 'programma di registr. su file sequenziale
1710 PRINT "R (MAIUSCOLO!)=registra";
1715 PRINT "    L=per leggere altro file"
1720 A$=INPUT$(1)
1730 IF A$="R" GOTO 1740 ELSE IF A$="L" THEN ERASE V$,F:GOTO 30
1735 GOTO 1250
1740 F1$="a:":INPUT "nome file =",F2$:F$=F1$+F2$
1750 M=ASC(F2$):IF (M<65 OR M>90) AND (M<96 OR M>122) GOTO 1740
1755 CLOSE 1
1760 OPEN "O",1,F$
1765 PRINT #1,RT:PRINT #1,CT
1767 D$=","
1770 FOR R=1 TO RT: FOR C=1 TO CT
1780 PRINT# 1, V$(F(R),C);D$
1790 NEXT C
1820 NEXT R: CLOSE
1830 GOTO 1250
1840 STOP
1850 GOTO 1250
1860 END
```

9.6 ARCHIVI AD ACCESSO DIRETTO

I file sequenziali, descritti nei paragrafi precedenti, sono impiegati soprattutto nei casi in cui i dati da registrare hanno una struttura molto semplice e libera.

Ad esempio, nella registrazione di una successione di numeri casuali, e' perfettamente inutile archiviarli in strutture piu' complesse di quella sequenziale; ma per archivi di dati organizzati secondo una certa struttura gerarchica e' opportuno impiegare i **file ad accesso diretto**, impropriamente chiamati random. Con questo tipo di archivi e' possibile specificare il singolo dato che si desidera leggere o scrivere.

Ovviamente, a operazioni piu' elaborate corrispondono anche istruzioni piu' complesse da comunicare alla macchina. La stessa istruzione OPEN di apertura di un file, e' diversa da quella dei file sequenziali e risulta cosi' articolata:

```
OPEN "R",NB,"a:ITALIA",NN
```

dove:

R e' il tipo di accesso Random

NB specifica il buffer che viene associato al trasferimento dei dati tra dischetto e memoria centrale

"a:ITALIA" e' l'identificatore del file, denominato ITALIA

NN e' l'ampiezza del record in caratteri.

Alla OPEN cosi' definita occorre associare l'istruzione **FIELD**, che ha la seguente struttura:

```
FIELD NB, Z1 AS R$ , Z2 AS P$ , Z3 AS C$     ...
```

dove:

NB e' un numero convenzionale che richiama una certa parte di memoria, chiamata buffer, messa a disposizione dalla macchina per gestire il trasferimento dei dati tra il dischetto e la memoria stessa;

Z1, Z2,.. sono le ampiezze, espresse in caratteri, dei vari campi in cui il record e' stato suddiviso, ciascuno dei quali porta un nome specificato alla destra di **AS** (in inglese **come**). In tal modo, ogni campo puo' contenere un aspetto, o attributo particolare del record.

Si voglia ora scrivere un programma per la registrazione su file ad

accesso diretto di un portafoglio clienti, costituito da una serie di nominativi con i relativi indirizzi, numeri di telefono e categoria. Il record sia, ad esempio, cosi' strutturato:

	categoria	**nome**	**telefono**	**indirizzo**
nome del campo	CAT	NOME	TEL	IND
FIELD	.....	....	....	....
ampiezza (car)	5	14	7	14

Segue il programma **prored** per la registrazione del record in un file ad accesso diretto.

```
10 'prored: registraz. su file ad accesso diretto.
20 CLS:KEY OFF
30 OPEN "R",1,"a:agenda",40
40 FIELD 1,5 AS CAT$,14 AS NOME$,7 AS TEL$,14 AS IND$
50 CLS
60 INPUT"R= ",R
70 INPUT"cat.= ",C$
80 INPUT"nome= ",N$
90 INPUT;"tel.= ",T$
100 INPUT"  ind= ",I$
110 LSET CAT$=C$
120 LSET NOME$=N$
130 LSET TEL$=T$
140 LSET IND$=I$
150 PUT# 1,R
160 INPUT"per continuare <a>-->",A$
170 IF A$="a" GOTO 50
180 CLOSE 1:END
```

Ecco le principali osservazioni:

1) la OPEN, rispetto all'analoga istruzione gia' vista per i file sequenziali,oltre all'indicazione "R", ora prevede anche l'indicazione della lunghezza del record;

2) i campi definiti nella 40 **devono essere associati a variabili stringa**, perche' il buffer accetta solo tali tipi di variabili;

3) la linea 60 specifica quanti sono i record da scrivere;

4) dalla linea 60 alla 100 sono state scritte le INPUT per l'assegnazione dei valori alle variabili esterne;
5) dalla 110 alla 140 i contenuti delle variabili esterne vengono riversate nelle variabili di campo (LSET);

6) alla linea 150 l'istruzione (**PUT**) che attiva lo scambio dati tra buffer e uno specifico record del file. Il numero di record puo' essere un numero intero compreso nell'intervallo 1... 32767 o un'espressione con un valore compreso in questo intervallo.

La figura 9.17 illustra il processo di registrazione.

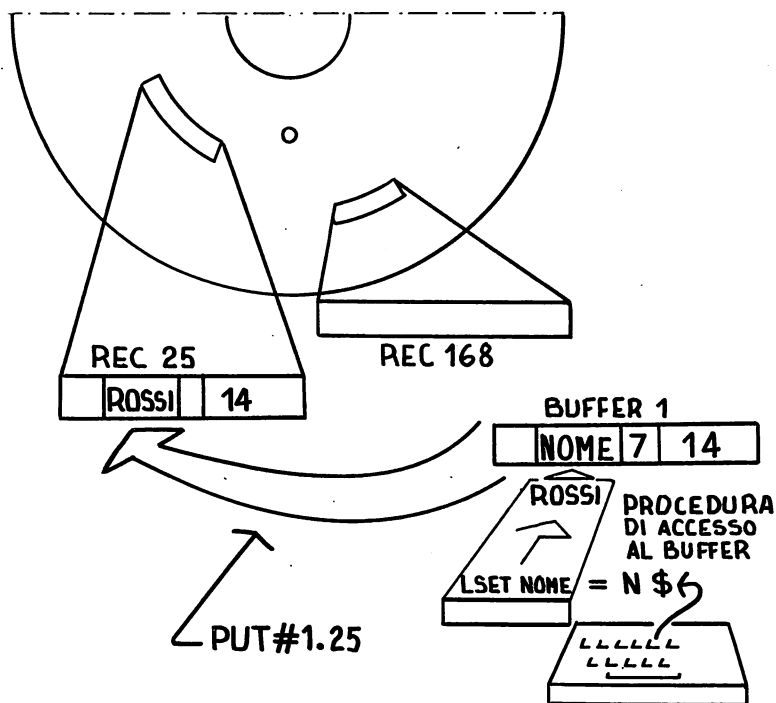



Figura 9.17

Simmetricamente alla PUT agisce la **GET**, che serve per leggere dati da un record registrato in un file ad accesso diretto e trasferirli in memoria centrale.

Come per la registrazione, occorre prima aprire il file specificando il metodo di accesso R (random). La lettura avviene dichiarando le variabili organizzate in campi secondo un'istruzione FIELD che dovra' definire un formato compatibile con quello dichiarato nell'operazione di registrazione.

Non e', invece, necessario utilizzare istruzioni tipo LSET.

Segue un programma per la lettura di un file random, denominato **leran.**

```
10 'leran: lettura di un file ad accesso diretto.
20 CLS:KEY OFF
30 OPEN "R",1,"a:agenda",40
40 FIELD 1,5 AS CAT$,14 AS NOME$,7 AS TEL$,14 AS IND$
60 INPUT"R= ",R
150 GET# 1,R
155 PRINT "nome: " NOME$, " categ.: "CAT$, " tel.: "TEL$
157 PRINT "ind.: " IND$
160 INPUT"per continuare <a>-->",A$
170 IF A$="a" GOTO 60
180 CLOSE 1:END
```

CAPITOLO
DIECI

I SEGRETI DI BOTTEGA

L'argomento dei "segreti" della bottega informatica ha un fondo comune a tutti gli elaboratori. La traduzione di questi segreti in regole pratiche e' quasi sempre legata ai piccoli particolari nascosti di ogni macchina. In questo capitolo abbiamo ripreso alcuni aspetti emersi nell'utilizzo di M24 e che possono rappresentare il bagaglio minimo di conoscenza da cui partire per ulteriori approfondimenti.

10.1 LA RIASSEGNAZIONE DEI TASTI FUNZIONALI

Come e' noto, ai tasti funzionali (F1÷F10) il sistema assegna il compito di generare una sequenza di caratteri di provata utilita'. Ad esempio, durante la messa a punto di un programma, quante volte per listarlo dovremmo digitare i 4 caratteri del comando basic "LIST" ? Ebbene, battendo F1, ogni volta risparmiamo 3 battute. Altrettanto dicasi per il tasto F2 al quale e' stato associato, oltre alla parola "RUN", anche il carattere AŞCII equivalente al tasto [CR].

E', tuttavia, possibile riassegnare le sequenze di caratteri secondo le necessita' dell'utilizzatore. Si noti che per mezzo del comando **key off** non si disabilitano i tasti funzionali, ma viene solo recuperata la venticinquesima riga del video. Se, invece,vogliamo far si' che ogni tasto funzionale generi una stringa personale (lunghezza massima 15 caratteri), e' sufficiente scrivere un'istruzione composta nel seguente modo:

```
key n,T$
```

dove **n** e' il numero d'ordine del tasto funzionale e T$ e' il nome di una stringa introdotta dall'esterno al programma, oppure generata dallo stesso programma.

Esempio: si voglia assegnare al tasto F5 il compito di generare la stringa **files" [CR]** basta scrivere

```
key 5,"files"+chr$(34)+chr$(13)
```

Si noti la necessita' di generare il carattere ["] mediante la funzione CHR$, anziche' metterlo in coda alla parola **files**: se cosi' avessimo fatto, saremmo incorsi in un errore di sintassi BASIC, a causa di una doppia notazione di chiusura di stringa. La funzione chr$(34) genera il carattere la cui entrata decimale 34 e' appunto ["], che viene accodato alla stringa files. Si ricordi che l'operatore [+] per le stringhe equivale a un'operazione di concatenamento.
Per **disabilitare rapidamente tutti i tasti funzionali** basta scrivere un ciclo cosi' fatto:

```
for I=1 to 10:key I,"":next      [""] equivale a stringa nulla.
```

Un aspetto di immediata utilita' per il programmatore e' quello di assegnare a un tasto funzionale il compito di salvare il programma caricato in memoria attraverso alcune istruzioni messe in testa al programma. Spieghiamoci con un esempio.

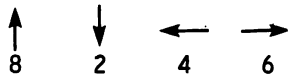
Si abbia un programma denominato **ted** e si voglia riassegnare al tasto **F4** il compito di generare la funzione di salvataggio del programma presente in memoria e non, genericamente, la stringa **save"**. Sara' sufficiente, tra le prime linee, scrivere un'istruzione di assegnazione di stringa, il cui nome compare nell'istruzione **key** successiva. La sequenza delle istruzioni e' semplicissima e appare costituita come segue:

```
10 'ted
20 T$=TED
30 KEY,4,"SAVE"+CHR$(34)+T$+CHR$(13)
 .
 .
```

In tal modo si e' certi che il salvataggio andra' a sovrapporre la versione attuale del programma su quella precedente conservata nel dischetto.

10.2 TASTI DEVIATORI DI PROGRAMMA

Anziche' funzioni generatrici di stringhe ai tasti funzionali e' possibile assegnare facolta' logiche: ad esempio, un programma puo' saltare a un sottoprogramma appena si preme un tasto al quale e' stata associata una data funzione. Queste facolta' possono essere estese anche ai quattro tasti che pilotano il cursore

↑ ↓ ← →
8 2 4 6

Le modalita' di assegnazione sono le seguenti:

ON KEY (N) GOSUB NL

dove **N** puo' avere valori compresi tra 1 e 14 (F1÷F14 + i 4 tasti frecce e **NL** e' il numero di linea al quale il programma salta se il tasto N e' stato premuto.

Il numero d'ordine delle frecce e' [11,14]=su,giu' e [12,14]=sinistra e destra.

Occorre fare attenzione a non confondere quanto detto con la manovra **KEY ON** che serve alla riattivazione della venticinquesima riga del video dedicata alla visualizzazione delle etichette dei tasti funzionali.

Dato che la funzione KEY (N) ON rallenta l'esecuzione del programma e' opportuno disattivarla quando non e' necessaria. La disattivazione si effettua con la manovra KEY (N) OFF.

10.3 IL CONTROLLO DELLA STAMPA

La stampante Olivetti Pr 15-B (equivalente alla OPE DM 5060) e' una stampante a punti progettata per produrre copie grafiche (hard-copy del contenuto del video) e per stampare testi. Come per analoghe stampanti, il funzionamento della PR 15 B puo' essere governato attraverso una predisposizione che definisce le densita' orizzontale e verticale. Questa definizione avviene mediante l'invio di caratteri speciali che non vengono stampati, ma che servono appunto a determinare le caratteristiche della stampa.

Ad esempio, per selezionare il funzionamento di stampa in larghezza doppia e' sufficiente premettere il messaggio **LPRINT CHR$(14).**

Per la stampa in grassetto, invece, bisogna premettere l'istruzione **LPRINT CHR$(15).** Il messaggio che fa tornare la stampante al funzionamento di partenza (10 caratteri per pollice) e': **LPRINT CHR$ (27).**

Per sperimentare completamente le possibilita' di stampa abbiamo realizzato una serie di programmi per l'analisi dei caratteri di governo. Il primo di questi e' **bast2**, di cui riportiamo il listato e il risultato di alcuni parametri di prova. La selezione avviene con l'introduzione di una coppia di caratteri di identificazione. Cosi' **11** sta per stampa tipo doppia larghezza; **cp** sta per stampa condensata, **e** sta per tipo esponente, **p** sta per tipo pedice (indice), cui corrispondono diverse densita' di stampa: queste vanno dalla micron (spaziatura 16.6 caratteri per pollice) alla doppia pica (5 caratteri per pollice). Sono possibili densita' intermedie: doppia elite (6.3), doppia micron (8), pica normale (10), elite (12). Per ciascuna densita', inoltre, e' possibile selezionare un'intensita' di battuta, a seconda delle passate degli aghi, di eventuale doppia battuta (grassetto) e con leggera scalatura orizzontale con risultati paragonabili a stampanti di qualita'.

```
10 'bast2:stampare in basic
20 CLS:KEY OFF
30 INPUT "TIPO DI STAMPA --> ",P$
40 IF P$="11" THEN  V$=CHR$(14):GOTO 90
50 IF P$="cp" THEN  V$=CHR$(15):GOTO 90
60 IF P$="gr" THEN  V$=CHR$(27)+CHR$(69):GOTO 90
70 IF P$="db" THEN  V$=CHR$(27)+CHR$(71):GOTO 90
80 IF P$="ml" THEN  V$=CHR$(15)+CHR$(14)
90 LPRINT P$;"  ";
100 LPRINT V$; "PROVA di stampa SU PR 15 B";CHR$(72);CHR$(20);CHR$(70);CHR$(
105 LPRINT'QQ$=INPUT$(1)
110 GOTO 30
```

```
cp  PROVA di stampa SU PR 15 BHF

gr  PROVA di stampa SU PR 15 BHF

db  PROVA di stampa SU PR 15 BHF

cp  PROVA di stampa SU PR 15 BHF

11  PROVA di stampa SU PR 15 BHF
```

Segue **bast5.**

```
10 'bast5:stampare in basic
20 CLS:KEY OFF
30 E$=CHR$(27)
40 INPUT;"x=",X
50 INPUT;" y=",Y
60 INPUT ;" z=",Z
70 INPUT " w=",W
80 V$=""
90 LPRINT CHR$(18);CHR$(20);E$+CHR$(70);E$+CHR$(72);E$+CHR$(84)
100 INPUT "TIPO DI STAMPA --> ",P$
110 IF P$="11" THEN  V$=CHR$(X):GOTO 180
120 IF P$="cp" THEN  V$=CHR$(Y):GOTO 180
130 IF P$="gr" THEN  V$=E$+CHR$(Z):GOTO 180
140 IF P$="db" THEN  V$=E$+CHR$(W):GOTO 180
150 IF P$="ml" THEN  V$=CHR$(X)+CHR$(Y)
160 IF P$="p" THEN  V$=E$+CHR$(83)+"0":GOTO 180
170 IF P$="e" THEN  V$=E$+CHR$(83)+"1":GOTO 180
180 LPRINT P$;"  ";
190 LPRINT V$; "PROVA di stampa SU PR 15 B";
200 GOTO 80
```

Alcune passate di bast5.

```
x=14 y=15 z=69 w=71

TIPO DI STAMPA --> 11
11  PROVA di stampa SU PR 15 B
TIPO DI STAMPA --> gr
gr  PROVA di stampa SU PR 15 B
TIPO DI STAMPA --> db
db  PROVA di stampa SU PR 15 B
TIPO DI STAMPA -->
RUN
x=14 y=15 z=77 w=80

TIPO DI STAMPA --> gr
gr  PROVA di stampa SU PR 15 B
TIPO DI STAMPA --> db
db  PROVA di stampa SU PR 15 B
TIPO DI STAMPA --> e
e   PROVA di stampa SU PR 15 B
TIPO DI STAMPA --> p
p   PROVA di stampa SU PR 15 B
```

E, infine, il programma **mist.**

```
10 'mist:stampare in basic
20 CLS:KEY OFF
30 DEFSTR E,P,V
40 P=CHR$(27)
50 LPRINT
60 INPUT "Larghezza= ",L
70 INPUT " Battuta= ",B
80 V1="":V2="":V3=""
85 IF L=15 THEN X=14 ELSE X=80:IF L=12 THEN X=77
90 IF B=0 THEN V1=P+CHR$(X):GOTO 150
```

```
100 IF B=1 THEN V1=P+CHR$(X):V2=P+CHR$(71):GOTO 160
110 IF B=2 THEN V1=P+CHR$(X):V2=P+CHR$(69):GOTO 170
120 IF B=3 THEN V1=P+CHR$(X):V2=P+CHR$(69):V3=P+CHR$(71):GOTO 180
130 IF B=4 THEN V1=P+CHR$(X):V2=P+CHR$(52):GOTO 190
140 IF B=5 THEN V1=P+CHR$(X):V2=P+CHR$(69):V3=P+CHR$(52):GOTO 200
150 LPRINT V1;"PROVA di stampa SU PR 15 B";:GOTO 50
160 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";P+CHR$(72);:GOTO 50
170 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";P+CHR$(70);:GOTO 50
180 LPRINT V1;V2;V3;"PROVA di stampa SU PR 15 B";P+CHR$(70);P+CHR$(72);:GOTO 5
190 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";:LPRINT;:GOTO 50
200 LPRINT V1;V2;V3;"PROVA di stampa SU PR 15 B";P+CHR$(70);P+CHR$(64);:GOTO 5
850 LPRINT V1;"PROVA di stampa SU PR 15 B";:GOTO 50
860 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";P+CHR$(72);:GOTO 50
870 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";P+CHR$(70);:GOTO 50
880 LPRINT V1;V2;V3;"PROVA di stampa SU PR 15 B";P+CHR$(70);P+CHR$(72);:GOTO 5
890 LPRINT V1;V2;"PROVA di stampa SU PR 15 B";:LPRINT;:GOTO 50
900 LPRINT V1;V2;V3;"PROVA di stampa SU PR 15 B";P+CHR$(70);P+CHR$(64);:GOTO 5
```

```
PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B

PROVA di stampa SU PR 15 B
PROVA di stampa SU PR 15 B

11  PROVA di stampa SU PR 15 B
gr  PROVA di stampa SU PR 15 B
db  PROVA di stampa SU PR 15 B

11  PROVA di stampa SU PR 15 B
gr  PROVA di stampa SU PR 15 B
db  PROVA di stampa SU PR 15 B
e   PROVA di stampa SU PR 15 B
p   PROVA di stampa SU PR 15 B
```

10.4 L'ASSEMBLAGGIO DI UN PROGRAMMA

Molti programmi esaminati in questo libro possono essere considerati come i mattoni di un programma piu' grande. Descriviamo, allora, come si procede per assemblare i vari "mattoni". L'istruzione Basic per effettuare questa fusione si chiama **MERGE** e ha una sintassi simile all'istruzione LOAD. L'unica differenza e' che, mentre il comando LOAD provoca l'azzeramento preliminare della memoria centrale, MERGE, invece, fa si' che il programma prelevato da dischetto si sovrapponga al programma in memoria. Questa sovrapposizione, pero', segue la regola generale secondo cui l'ultimo numero di linea introdotto vince e cancella il precedente. Pertanto, se vogliamo far coesistere il programma in memoria con quello chiamato, i numeri di linea dei due programmi devono cadere in intervalli diversi, o, comunque, senza ricoprimenti indesiderati. Inoltre, il programma chiamato con il MERGE deve essere stato registrato in ASCII, cioe' con un comando SAVE, opzione A (vedi oltre al punto 2).

Esaminiamo, ora, un semplice programma di **fusione.** Supponiamo di voler fondere i seguenti programmi di nome R e S gia' registrati nel dischetto in b.

```
10 'R                          10 'S
20 PRINT "Nel mezzo"           20 PRINT "di nostra"
30 PRINT "del cammin"          30 PRINT "vita"
40 END                         40 END
```

Per fondere i due programmi in un unico programma che risulti cosi' articolato:

```
10 ' R e S
20 PRINT "Nel mezzo"
30 PRINT "del cammin"
40 PRINT "di nostra"
50 PRINT "vita"
60 END
```

occorre procedere effettuando i seguenti passi.

1) Caricamento in memoria di R con LOAD"b:R [CR].

2) Salvataggio di R su dischetto con registrazione in codice ASCII, mediante il comando SAVE"b:R.BAS",A [CR].

3) Caricamento in memoria di S con LOAD"b:S [CR].

4) Rinumerazione di S a partire da 1000 (un numero di linea sufficientemente alto che sia al di la' dell'estremo superiore di R), mediante il comando RENUM 1000 [CR].

5) Fusione di R su S con il comando MERGE"b:R [CR].

Il programma risultante avra' la seguente successione di numeri di linea:

10 20 30 40 1000 1010 1020 1030

Cancelliamo le linee 40 (END) e 1000 ('S) e modifichiamo la 1Q in modo che il commento risulti: 'R e S. Impostiamo, infine, renum [CR] e otterremo il programma desiderato.

10.5 IL CONCATENAMENTO DEI PROGRAMMI

Quando un programma e' piu' grande della memoria centrale disponibile, sorge la necessita' di ricercare un compromesso tra possibilita' di esecuzione e prestazioni risultanti. Supponiamo che il programma in questione non sia un unico blocco, ma sia stato progettato in sezioni da attivare con la tecnica delle istruzioni tipo gosub e return. Anche in questo caso, naturalmente, se le dimensioni complessive superano la capacita' di memoria, non sara' possibile farlo girare: grazie alla sua struttura, pero', sara' possibile aggirare l'ostacolo. Infatti, basta avere un meccanismo software che, anziche' utilizzare le gosub, attivi una lettura da dischetto di un particolare **segmento** di programma. Questo costituira' un'appendice di quello principale in memoria. Il meccanismo viene realizzato nel Basic M24 con la coppia di istruzioni **CHAIN** e **COMMON**.
Si abbiano ora i seguenti programmi, denominati INTER e MILAN.

```
10 'Milan
20 INPUT "A= ",A
30 PRINT A
40 COMMON A
50 CHAIN"b:Inter
60 END
```

```
10 'Inter
20 INPUT "B= ",B
30 PRINT B
40 COMMON A
50 PRINT A+B
60 END
```

Il procedimento e' illustrato nella figura 10.1: il programma chiamante ha ceduto il posto al programma chiamato passandogli anche i valori delle variabili in comune.

Il programma Inter, registrato su dischetto, viene portato in memoria dall'istruzione CHAIN scritta alla linea 50 del programma Milan, che, inizialmente, e' stato lanciato dall'operatore col comando RUN. Quando si esegue l'istruzione 50 [chain"b:Inter], il programma Inter viene chiamato da dischetto in memoria centrale; il valore attuale della variabile A, dichiarato nella COMMON di Milan alla linea 40, viene passato all'Inter, grazie alla COMMMON che e' stata scritta alla linea 40 di tale programma; in tal modo, durante l'esecuzione dell'Inter, la variabile A e' operabile con il suo contenuto attuale, come il Milan l'aveva assunta. Una COMMON puo' contenere piu' variabili (separate da virgole) e uno stesso programma puo' contenere piu' COMMON.

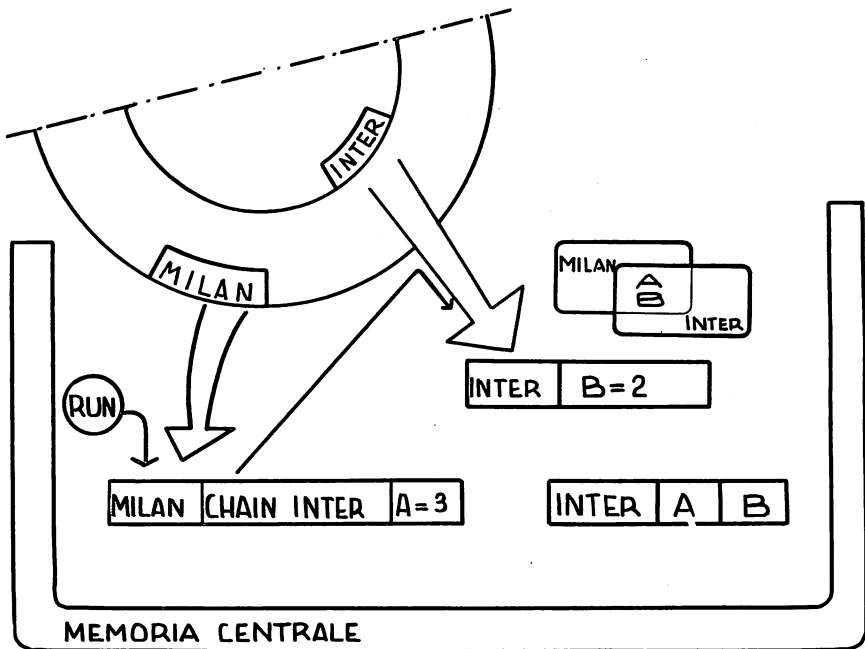



Figura 10.1

Se i due programmi da concatenare hanno **tutte** le variabili in comune e non soltanto un sottoinsieme, si puo' evitare di dichiararle in entrambi con la COMMON: bastera' scrivere **all** [tutte] nell'istruzione chain, che risultera', quindi, cosi' strutturata:

```
NL CHAIN"b:Inter,NL1,ALL
```

dove:

NL e' il numero di linea in cui e' stata scritta l'istruzione CHAIN;

NL1 e' l'eventuale numero di linea della prima istruzione che deve essere eseguita nel programma Inter.

Se NL1 manca, l'Inter viene eseguito alla linea indicata dalla sua prima istruzione.

Il caso interessa quei programmi troppo lunghi che devono essere spezzati in due o piu' segmenti da far eseguire consecutivamente.
Nell'esempio appena trattato la struttura del programma deve tener conto del fatto che, a parte le variabili, una volta che un programma ha invaso la memoria, l'altro viene automaticamente cancellato.
Se, invece di un concatenamento dinamico di programmi, o segmenti di programma come nei casi appena trattati, vogliamo farne una **fusione** permanente, occorre procedere come e' illustrato alla figura 10.2.

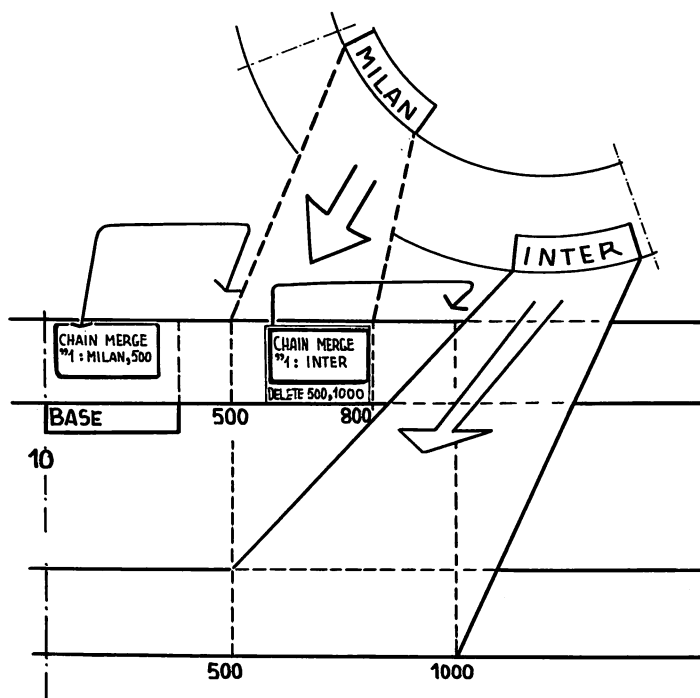



Figura 10.2

Si abbiano i due soliti programmi Milan e Inter registrati su dischetto e un programma di lancio iniziale, che chiamiamo Base. La procedura e' la seguente: Base chiama Milan che giri dall'istruzione 500; a sua volta, prima di terminare, Milan chiama Inter che giri dall'istruzione 500, con preventiva cancellazione della memoria dalla linea 500 alla 1000.

UN LABORATORIO PER ESPERIMENTI MATEMATICI

L'efficacia di uno strumento matematico utilizzato per scopi tecnici e operativi si misura essenzialmente mediante una serie di prove pratiche fatte analizzando fenomeni rappresentabili opportunamente nei limiti della validita' dello strumento e che altrimenti non sarebbe altrettanto facile studiare. Questa appendice non ha alcuna pretesa di voler trattare esaurientemente gli argomenti scelti, ma vuol fornire solo un pretesto per verificarne la conoscenza: il fatto nuovo, semmai, sta nel rivisitare i contenuti del singolo strumento matematico esprimendone la struttura in linguaggio informatico, e una volta realizzato il programma equivalente alla formalizzazione adottata, cioe', una volta tradotto in Basic l'algoritmo, provare a sperimentare le relazioni tra enti e valori di un certo fenomeno che vogliamo studiare. Lasciamo agli utilizzatori la scelta delle classi di applicazioni cui rivolgere i programmi qui trattati, ricordando, tuttavia, che alcuni di essi sono i principali operatori per esprimere in modo sufficientemente sintetico fenomeni elettrici, strutture di scienza delle costruzioni, strutture economiche, etc.

A.1 LE MATRICI

A.1.1 ALCUNI RICHIAMI FONDAMENTALI

Una matrice e' un insieme ordinato di numeri disposti per righe e per colonne in modo da formare una cornice rettangolare. Si dice matrice di ordine (r,c) quella che ha i numeri disposti ordinatamente in **r** righe e in **c** colonne. Si conviene, inoltre, di chiamare con a_{rc} o a(r,c) l'elemento all'incrocio della riga r e della colonna c.

Indicheremo con **A** la matrice rappresentata nel seguente modo:

$$A = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1c} \\ a_{21} & a_{22} & \cdots & a_{2c} \\ . & . & & . \\ a_{r1} & a_{r2} & \cdots & a_{rc} \end{vmatrix}$$

Gli elementi a(r,c) di **A** per cui r=c si chiamano elementi **diagonali** di **A.** Inoltre, se la matrice e' quadrata (r=c) e tutti gli elementi sono nulli eccetto i diagonali (quelli per cui r=c), tale particolare matrice quadrata si dice **diagonale.** Se, in tale matrice diagonale, tutti gli elementi diagonali sono unitari, questa particolare matrice diagonale si chiama **unitaria** o matrice identita'.

La **trasposta** di una matrice e' una matrice ottenuta scambiando le righe con le colonne della matrice data.

A.1.2 ELEMENTI DI CALCOLO MATRICIALE

Prima di operare sulle matrici ricordiamo quanto abbiamo gia' appreso nell'esercizio di programmi come **restab2** in cui, dichiarando il numero delle righe e delle colonne, si puo' generare una tabella (matrice) e registrarla in un file sequenziale con un certo nome. Introducendo il nome e operando con programmi tipo **lestab2** e' possibile riportare su video, per eventuali ulteriori trattamenti, matrici registrate.

Il seguente programma, denominato **tramat**, consente di generare una matrice di **r** righe e di **c** colonne e di produrne la trasposta. Il programma, informaticamente parlando, e' al limite della semplicita': non abbiamo pertanto introdotto protezioni contro introduzioni errate e altre sofisticazioni che avrebbero reso meno evidente e semplice la lettura del listato. Naturalmente il lettore puo' rivederne il nucleo e farne la sua versione personale, amichevole e protetta da ogni incauta manovra.

```
L10 'tramat:.........trasposta di una matrice
20 CLS:KEY OFF
30 PRINT "LA TRASPOSTA DI UNA MATRICE":PRINT
40 INPUT;"n. rig = ",RT
50 INPUT "   n. col = ",CT:PRINT
60 FOR R=1 TO RT  'gen. mat. di ordine (r,c)
70 FOR C=1 TO CT
80 PRINT "   V("R","C")=";
90 INPUT;"",V(R,C)
100 NEXT C:PRINT :NEXT R
110 PRINT :PRINT "ECCO LA MATRICE INTRODOTTA"
120 FOR R=1 TO RT'visual. mat.(r,c) generata
130 FOR C=1 TO CT
140 PRINT "  V("R","C")="V(R,C);
150 NEXT C:PRINT:NEXT R
160 PRINT :PRINT "......E LA SUA TRASPOSTA"
170 PRINT
180 FOR R=1 TO RT 'visualizz. tra.mat.(c,r)
190 FOR C=1 TO CT
200 PRINT V(C,R);
210 MT(R,C)=V(C,R)'salvat.tramat in MT(r,c)
220 NEXT:PRINT:NEXT:PRINT
230 FOR R=1 TO RT 'vis. MT(r,c)
240 FOR C=1 TO CT
250 PRINT MT(R,C);
260 NEXT C:PRINT:NEXT R
RUN
LA TRASPOSTA DI UNA MATRICE

n. rig = 3   n. col = 3

   V( 1 , 1 )=1   V( 1 , 2 )=2   V( 1 , 3 )=3
   V( 2 , 1 )=4   V( 2 , 2 )=5   V( 2 , 3 )=6
   V( 3 , 1 )=7   V( 3 , 2 )=8   V( 3 , 3 )=9
```

```
ECCO LA MATRICE INTRODOTTA
  V( 1 , 1 )= 1   V( 1 , 2 )= 2   V( 1 , 3 )= 3
  V( 2 , 1 )= 4   V( 2 , 2 )= 5   V( 2 , 3 )= 6
  V( 3 , 1 )= 7   V( 3 , 2 )= 8   V( 3 , 3 )= 9

......E LA SUA TRASPOSTA

 1  4  7
 2  5  8
 3  6  9

 1  4  7
 2  5  8
 3  6  9
```

Si noti che, nella passata del programma, si stampano due matrici identiche, trasposte di quella data. Sono identiche come risultato, ma l'algoritmo per ottenerle e' diverso . Infatti la prima proviene dal ciclo 180÷220 in cui l'elemento generico della matrice ha un argomento con gli indici scambiati rispetto a quello utilizzato per la generazione (linea 90); la seconda stampa e' stata ottenuta, invece, col ciclo 230÷260 in cui l'elemento generico MT(r,c) e' la copia di V(c,r) ottenuta alla linea 210.

Una variante di tramat e' **tramat1** che opera su una matrice registrata con restab2: si possono, cosi', fare infinite prove senza la noia di introdurre la matrice di partenza. Possiamo, ovviamente, fabbricare una collezione di matrici registrate ciascuna col suo nome. La variante introdotta per leggere la matrice da un file esterno puo' essere considerata un'opzione di utilita' generale per tutti i programmi che seguono e che contengono una sezione introduttiva per la generazione di una matrice, Si lascia al lettore il compito di apporre tale variante. Come indicazione, segue tramat1.

```
10 'tramat1:........trasposta di una matrice
20 CLS:KEY OFF
30 PRINT "LA TRASPOSTA DI UNA MATRICE";
40 INPUT;" *** NOME DELLA MATRICE=",M$:PRINT
50 OPEN "I",1,M$
52 INPUT #1,RT:INPUT #1,CT' estrae le dimen-
53 'sioni della matr.(RT=max rig.;RC=max col)
60 FOR R=1 TO RT 'legge mat. di ordine (r,c)
70 FOR C=1 TO CT
90 INPUT #1,V(R,C)
100 NEXT C:NEXT R:PRINT
110 PRINT "ECCO LA MATRICE REGISTRATA":PRINT
120 FOR R=1 TO RT'visual. mat.(r,c) registr.
130 FOR C=1 TO CT
140 PRINT "  V("R","C")="V(R,C);
150 NEXT C:PRINT:NEXT R
160 PRINT :PRINT "......E LA SUA TRASPOSTA"
170 PRINT
180 FOR R=1 TO RT 'visualizz. tra.mat.(c,r)
```

```
190 FOR C=1 TO CT
200 PRINT V(C,R);
210 MT(R,C)=V(C,R)'salvat.tramat in MT(r,c)
220 NEXT:PRINT:NEXT:PRINT
230 FOR R=1 TO RT 'vis. MT(r,c)
240 FOR C=1 TO CT
250 PRINT MT(R,C);
260 NEXT C:PRINT:NEXT R
RUN
LA TRASPOSTA DI UNA MATRICE *** NOME DELLA MATRICE=qw

ECCO LA MATRICE REGISTRATA

  V( 1 , 1 )= 1   V( 1 , 2 )= 2   V( 1 , 3 )= 3
  V( 2 , 1 )= 4   V( 2 , 2 )= 5   V( 2 , 3 )= 6
  V( 3 , 1 )= 7   V( 3 , 2 )= 8   V( 3 , 3 )= 9
```

SOMMA DI MATRICI

Il seguente programma **somat** esegue la somma di due matrici M1 e M2, di uguale ordine, introdotte da tastiera. Come e' noto, tale somma si ottiene costruendo una matrice (MS) i cui elementi MS(r,c) sono somma dei corrispondenti elementi di M1 e M2.

```
10 'somat:.....              somma di due matrici
20 CLS:KEY OFF
30 PRINT "LA SOMMA DI DUE MATRICI ***       ";
32 INPUT;"n.righe=",RT
35 INPUT "         n.colonne=",CT:PRINT
37 PRINT "INTRODURRE LA PRIMA MATRICE"
40 FOR R=1 TO RT 'genera M1 di ordine (RT,RC)
45 FOR C=1 TO CT
47 PRINT "  M1("R","C")=";
50 INPUT;"",M1(R,C)
55 NEXT:PRINT :NEXT:PRINT:PRINT
57 PRINT "INTRODURRE LA SECONDA MATRICE"
60 FOR R=1 TO RT 'genera M2 di ordine (RT,RC)
70 FOR C=1 TO CT
77 PRINT "  M2("R","C")=";
90 INPUT;"",M2(R,C)
100 NEXT:PRINT:NEXT:PRINT
110 PRINT "ECCO LA MATRICE RISULTANTE":PRINT
120 FOR R=1 TO RT'visual. mat.(r,c) registr.
130 FOR C=1 TO CT
140 PRINT "  MS("R","C")="M1(R,C)+M2(R,C);
150 NEXT C:PRINT:NEXT R
160 END:' :PRINT "INTRODURRE LA SECONDA M."
170 PRINT
180 FOR R=1 TO RT 'visualizz. tra.mat.(c,r)
190 FOR C=1 TO CT
200 PRINT "  M1("R","C")=";
205 INPUT ;"",M1(R,C)
210 MS(R,C)=V(R,C)+M1(R,C)'somma le matrici
```

```
220 NEXT:PRINT:NEXT:PRINT
230 FOR R=1 TO RT 'vis. Ms(r,c)
240 FOR C=1 TO CT
250 PRINT MS(R,C);
260 NEXT C:PRINT:NEXT R
```

Segue la variante **somat1** che preleva la prima matrice da dischetto dichiarandone il nome; la seconda, invece, va introdotta da tastiera.

```
10 'somat1:...somma di due matrici di cui la
20 CLS:KEY OFF'     prima registrata su file
30 PRINT "LA SOMMA DI DUE MATRICI ***  NOME";
40 INPUT;" DELLA PRIMA MATRICE=",M$:PRINT
50 OPEN "I",1,M$
52 INPUT #1,RT:INPUT #1,CT' estrae le dimen-
53 'sioni della matr.(RT=max rig.;RC=max col)
60 FOR R=1 TO RT 'legge mat. di ordine (r,c)
70 FOR C=1 TO CT
90 INPUT #1,V(R,C)
100 NEXT C:NEXT R:PRINT
110 PRINT "ECCO LA MATRICE REGISTRATA":PRINT
120 FOR R=1 TO RT'visual. mat.(r,c) registr.
130 FOR C=1 TO CT
140 PRINT "  V("R","C")="V(R,C);
150 NEXT C:PRINT:NEXT R
160 PRINT :PRINT "INTRODURRE LA SECONDA M."
170 PRINT
180 FOR R=1 TO RT 'visualizz. tra.mat.(c,r)
190 FOR C=1 TO CT
200 PRINT "  M1("R","C")=";
205 INPUT ;"",M1(R,C)
210 MS(R,C)=V(R,C)+M1(R,C)'somma le matrici
220 NEXT:PRINT:NEXT:PRINT
225 PRINT"ECCO LA SOMMA DELLE MATRICI":PRINT
230 FOR R=1 TO RT 'vis. Ms(r,c)
240 FOR C=1 TO CT
250 PRINT MS(R,C);
260 NEXT C:PRINT:NEXT R:PRINT:LOCATE 22,50
270 INPUT "battere <CR> per continuare",QQ
280 GOTO 20
```

MOLTIPLICAZIONE DI MATRICI

Prima di presentare il programma per la moltiplicazione di due matrici e' opportuno tener presenti le seguenti definizioni e regole preliminari:

1) Il prodotto di uno scalare k (un numero in senso elementare, in contrapposizione a vettori e matrici) per una matrice **A** e' la matrice che si ottiene da **A** moltiplicando per k tutti gli elementi:

$$k[a(r,c)]=[ka(r,c)] .$$

Per sperimentare questa definizione basta, ad esempio, riprendere

il programma **tramat** e, se k=2, modificare la linea 210, in modo che diventi:

```
210  MT(R,C)=2*V(R,C)'salv........
```

L'istruzione di assegnazione (o di duplicazione della matrice **V** nella matrice **MT**) e' ora un'assegnazione con modifica (il prodotto per il fattore 2) di ogni elemento della matrice **V** nella matrice **MT**.

2) Si chiama **prodotto interno** di un vettore riga **R** di ordine **c** (matrice di una sola riga costituita di c elementi) per un vettore colonna **C** di ordine **c** (matrice di una sola colonna di c elementi), il numero

$$[r_1, r_2, \dots, r_c] \begin{vmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_c \end{vmatrix} = r_1c_1 + r_2c_2 + \dots + r_cc_c \, .$$

Con queste definizioni, estendendo il concetto di prodotto interno dai vettori alle matrici, si trae anche, convenzionalmente, la definizione di moltiplicazione tra matrici, dove i fattori, presi con opportune regole, sono le righe dell'una e le colonne dell'altra matrice. Si deve, innanzitutto osservare, che la moltiplicazione tra due matrici ha significato solo se **la seconda matrice ha tante righe quante sono le colonne della prima.** Da cio' discende, in generale, che la moltiplicazione tra matrici non gode della proprieta' commutativa. Supponiamo, allora, di avere due matrici, **F**(3,5) e **G**(5,4); il loro prodotto e' possibile perche' la prima ha tante colonne (5) quante sono le righe della seconda. Il loro prodotto **FxG** e' una matrice con 3 righe e 4 colonne. Vediamo il dettaglio delle operazioni, illustrate nella figura A.1.

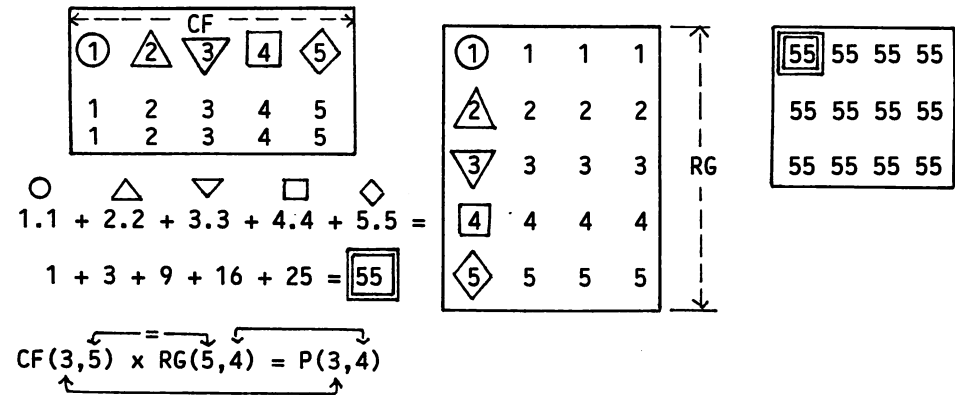



Figura A.1

Simbolicamente, siano

$$F = \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ f_{31} & f_{32} \end{vmatrix} \quad ; \quad G = \begin{vmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{vmatrix} . \text{ Sara'}$$

$$P = F \times G = \begin{vmatrix} f_{11}g_{11} + f_{12}g_{21} & f_{11}g_{12} + f_{12}g_{22} \\ f_{21}g_{11} + f_{22}g_{21} & f_{21}g_{12} + f_{22}g_{22} \\ f_{31}g_{11} + f_{32}g_{21} & f_{32}g_{12} + f_{32}g_{22} \end{vmatrix}$$

Dall'esempio simbolico appena illustrato si nota che gli elementi in colonna della matrice G si abbinano con quelli della F organizzati in binomi per colonna di matrice prodotto. **Da questa procedura di calcolo si induce l'intreccio** dei tre **cicli annidati nelle linee 220÷250 del seguente programma** molgen **per il calcolo del prodotto di due matrici.**

```
10 'molgen: prodotto di matrici
20 CLS:KEY OFF:PRINT"IL PRODOTTO DI DUE MATRICI"
30 PRINT :INPUT;"Righe di F = ",RF
40 INPUT ";   Colonne di F = ",CF
50 INPUT;"Righe di G = ",RG
60 INPUT ";   Colonne di G = ",CG
70 PRINT :IF RG<>CF THEN 20
80 FOR R=1 TO RF
90 FOR C=1 TO CF
100 PRINT "   F("R","C")=";
110 INPUT;"",F(R,C):NEXT:PRINT:NEXT:PRINT
120 FOR R=1 TO RG
130 FOR C=1 TO CG
140 PRINT "   G("R","C")=";
150 INPUT;"",G(R,C):NEXT:PRINT:NEXT:PRINT
160 FOR R=1 TO RF
170 FOR C=1 TO CF
180 PRINT F(R,C);:NEXT:PRINT:NEXT:PRINT
190 FOR R=1 TO RG
200 FOR C=1 TO CG
210 PRINT G(R,C);:NEXT:PRINT:NEXT
220 FOR C=1 TO CG
230 FOR R=1 TO RF
240 FOR Q=1 TO CF
250 P(R,C)=P(R,C)+F(R,Q)*G(Q,C):NEXT:NEXT:NEXT:PRINT
260 PRINT"LA MATRICE PRODOTTO:":PRINT
270 FOR R=1 TO RF
280 FOR C=1 TO CG
290 PRINT P(R,C);:NEXT:PRINT:NEXT:END
Ok_
RUN
IL PRODOTTO DI DUE MATRICI

Righe di F = 3;   Colonne di F = 3
Righe di G = 3;   Colonne di G = 3

   F( 1 , 1 )=5   F( 1 , 2 )=5   F( 1 , 3 )=3
   F( 2 , 1 )=3   F( 2 , 2 )=5   F( 2 , 3 )=5
   F( 3 , 1 )=2   F( 3 , 2 )=3   F( 3 , 3 )=4

   G( 1 , 1 )=400   G( 1 , 2 )=600   G( 1 , 3 )=200
   G( 2 , 1 )=300   G( 2 , 2 )=400   G( 2 , 3 )=100
   G( 3 , 1 )=250   G( 3 , 2 )=400   G( 3 , 3 )=150

 5  5  3
 3  5  5
 2  3  4

 400  600  200
 300  400  100
 250  400  150

LA MATRICE PRODOTTO:

 4250  6200  1950
 3950  5800  1850
 2700  4000  1300
Ok_
```

Come esempio di applicazione del prodotto di due matrici, supponiamo di avere 3 progetti T1, T2, e T3 e 3 tipi di figure professionali U1, U2 e U3 e di voler conoscere, date le rispettive giornate lavorate (matrice F),dati i costi (C), i ricavi (R) e i profitti (R-C) della giornata lavorata dalla singola figura professionale (matrice G), quali siano i costi, i ricavi e i profitti dei vari progetti. Siano, quindi

F(3,3) =

	U1	U2	U3
T1	5	5	3
T2	3	5	5
T3	2	3	4

; **G(3,3)** =

	C	R	R-C
U1	400	600	200
U2	300	400	200
U3	250	400	150

;

P = F x G =

	ΣC	ΣR	Σ(R-C)
T1	4250	6200	1950
T2	3950	5800	1850
T3	2700	4000	1300

Un caso interessante di moltiplicazione tra matrici si verifica quando il prodotto di due matrici quadrate dello stesso ordine da' per risultato la matrice unitaria o **identita'** (tutti gli elementi sono nulli, eccetto quelli diagonali del tipo a(1,1), a(2,2), ... a(n,n) che sono unitari): si ha, quindi, lo stesso risultato del prodotto di un numero per il suo reciproco. Per analogia si conviene di chiamare le due matrici l'una **inversa** dell'altra.

A.1.3 OPERAZIONI ELEMENTARI SULLE MATRICI

Prima di procedere oltre, e' opportuno conoscere alcune proprieta' elementari del calcolo matriciale in base alle quali e con opportune combinazioni di esse si puo' operare sulle righe e sulle colonne di una matrice **senza alterarne il valore.** Per scopi didattici, riportiamo gli enunciati delle operazioni piu' importanti che vengono utilizzate nell'algoritmo per il calcolo della matrice inversa.

1) Moltiplicare gli elementi di una determinata riga per una costante (scalare) non nulla.

2) Sommare due righe e sostituire una di esse con la somma.

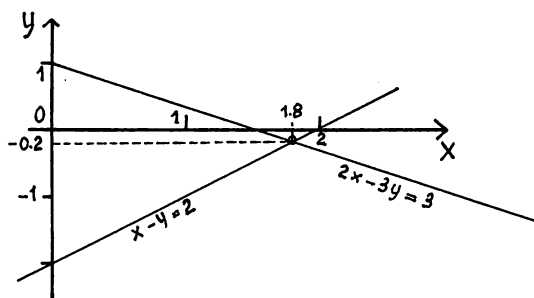
3) Fare una combinazione delle operazioni 1) e 2), cioe' sommare agli elementi di una riga i corrispondenti elementi di un'altra moltiplicati per un fattore costante.

Facciamo un esempio del terzo tipo con una matrice composta dalla matrice dei coefficienti delle incognite e dal vettore colonna corrispondente ai termini noti del seguente sistema di due equazioni in due incognite:

$$\begin{cases} 2x + 3y = 3 \\ \ x - \ y = 2 \end{cases}$$

Sappiamo che il sistema ammette una soluzione se il determinante e' diverso da zero; infatti

$$\begin{vmatrix} 2 & 3 \\ 1 & -1 \end{vmatrix} = (-1)(2) - (3) = -5$$

Nel piano XY la soluzione e' data dal punto comune delle due rette (figura A.2).

x	y
0	1
1.5	0

x	y
0	-2
2	0

Figura A.2

La soluzione analitica per sostituzione, come e' noto, si svolge cosi': dalla **x-y=2** si ricava **x=y+2** che viene sostituito nella **2x+3y=3.** Si ha allora:

2(y+2)+3y=3, da cui si ricava la y.

2y+4+3y=3 ; 5y=-1, **y=-1/5=-.2**

Sostituendo il valore nella **x-y=2** si ha

x=y+2=-1/5+2=9/5=1.8 **x=1.8**

Tornando alle operazioni elementari, eseguendo il primo tipo di operazione, moltiplichiamo la seconda equazione per il fattore **-2** e il sistema diventa:

$$\begin{cases} 2x + 3y = 3 \\ -2x + 2y = -4 \end{cases}$$

Col secondo tipo di operazioni sostituiamo al posto di **x -y =2** la somma ottenuta sommando membro a membro le due equazioni del sistema, cioe' [0 + 5y=-1].

Il sistema dato si e' cosi' trasformato in quello equivalente.

$$\begin{cases} 2x + 3y = 3 \\ \quad\;\; 5y = -1 \end{cases} .$$

Nella seconda equazione di tale sistema non compare piu' l'incognita **x**, ma la soluzione del sistema non e' cambiata anche se la sua rappresentazione geometrica e' ora quella illustrata nella figura A.3.

x	y
0	1
1.5	0

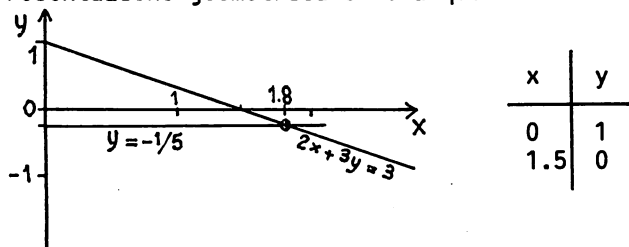



Figura A.3

Il metodo, detto di eliminazione, puo' essere esteso alla soluzione di un sistema di **n** equazioni in **n** incognite. Ad esempio, puo' essere applicato al seguente sistema di tre equazioni in tre incognite

$$\begin{cases} 2x + 2y + 2z = 1 \\ \;\; x - \;\; y + \;\; z = 1 \\ \qquad\qquad 2z = 1 \end{cases} \quad \text{porta alla soluzione} \quad \begin{cases} x = 1/4 \\ y = -1/4 \\ z = 1/2 \end{cases} ,$$

come e' illustrato anche nella rappresentazione di figura A.4, in cui le tre equazioni sono tre piani e la soluzione e' costituita dalle coordinate del loro punto comune.

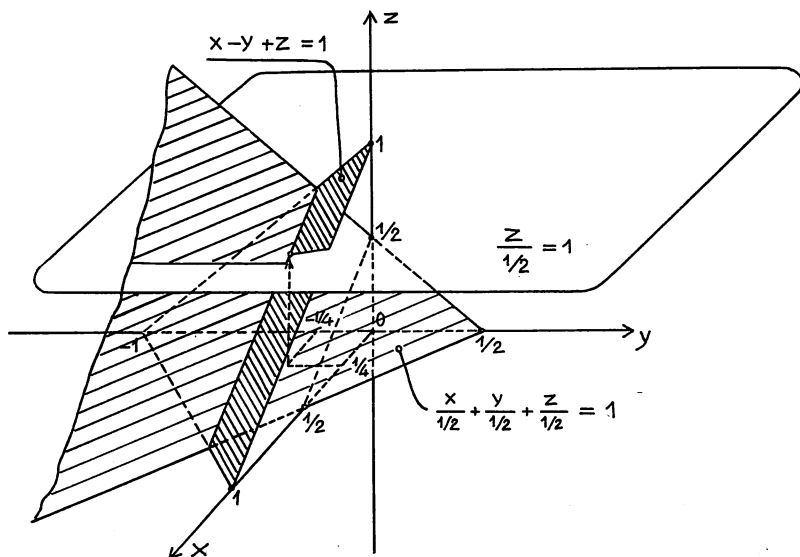



Figura A.4

Le operazioni elementari applicate ai sistemi di equazioni trovano corrispondenza nel calcolo matriciale, dove le proprieta' si arricchiscono di situazioni interessanti sul profilo pratico. Vediamo nel programma **giomin3** un algoritmo che serve a trasformare una matrice quadrata di D x D elementi, **di cui nessuno nullo lungo la diagonale** (a(r,c) con r=c), in quella relativa a un sistema di equazioni equivalente in cui, salvo quelli nella diagonale suddetta, tutti gli altri sono stati ridotti a zero con opportune operazioni elementari sulle righe.

```
10 'giomin3
20 CLS:KEY OFF:PRINT"CALCOLO DELLA MATRICE DIAGONALE   -";
22 PRINT CHR$(16)"PASSO A PASSO (s/n)   -";CHR$(16);
23 PAP$=INPUT$(1):IF PAP$<>"s" AND PAP$<>"n" THEN 20
25 INPUT "DIMENS. MATRICE=",D$:D=VAL(D$):IF D=0 THEN 20
50 PRINT:FOR R=1 TO D:FOR C=1 TO D
66 PRINT "  a("R","C")=";:INPUT;"",A(R,C):PRINT";";
70 NEXT:PRINT:NEXT:PRINT
75 PRINT "LA MATRICE APPENA INTRODOTTA"
80 FOR R=1 TO D:FOR C=1 TO D:PRINT A(R,C);
120 NEXT:PRINT:NEXT:PRINT
130 FOR Q=1 TO D' operazioni elementari sulle righe
135 FOR R=1 TO D
136 IF A(Q,Q)=0 THEN  PRINT"A'("Q","Q")=0":END
139 M=-A(R,Q)/A(Q,Q)
140 FOR C=1 TO D
160 IF Q=R THEN M=0:GOTO 180
180 A(R,C)=M*A(Q,C)+A(R,C)
182 IF PAP$="n" THEN 200 'esclusione del passo a p.
185 PRINT "m("R","Q")=-a("R","Q")/a("Q","Q")="M;
187 PRINT"  --";CHR$(16);"a("R","C")=";A(R,C)
195 QQ$=INPUT$(1):GOSUB 1000
200 NEXT:NEXT:NEXT:FF=1
1000 'stampa matrice---------------------------------
1025 IF FF=1 THEN 1050
1027 IF PAP$="s" THEN 1100
1050 PRINT CHR$(16);CHR$(16);" LA MATRICE DIAGONALE"
1100 IF FF=0 THEN PRINT:PRINT"LA MATRICE INTERMEDIA"
1230 FOR S=1 TO D:FOR T=1 TO D
1242 Y=A(S,T)
1243 IF Y=0 THEN W=1:GOTO 1250
1245 IF ABS(Y)<.000001 THEN  PRINT " 0";:W=1:GOTO 1260
1250 PRINT A(S,T);
1260 NEXT:PRINT:NEXT:PRINT:IF W=1 AND FF=1 THEN END
1280 RETURN
```

Si noti che la scelta del PASSO A PASSO (linea 22) consente di vedere la trasformazione della matrice durante le varie operazioni eseguite dall'assegnazione alla linea 180, che aggiorna dinamicamente tutti gli elementi della matrice data fino a ridurla alla forma **diagonale.** La formula alla linea 139 (che calcola il fattore m) e' stata messa prima del ciclo delle colonne in modo che il valore di **m** valga per tutta la riga. Per evitare risultati del tipo 1.E-13 che occorrono al posto dello zero quando intervengono errori di arrotondamento, e'

stata posta la condizione alla linea 1245 che ripristina lo zero allorche' il risultato e' inferiore a 0.0000001. Per evitare di azzerare gli elementi diagonali che porterebbero a situazioni di errore (division by zero), e' stata posta la linea 160, la quale, rivelando che Q=R (cioe' la riga Q di trasformazione porta lo stesso numero d'ordine della riga su cui operare), annulla il coefficiente **m** calcolato dalla 139 e, quindi, evita l'azzeramento della 180. Questa con m=0 si limita, quindi, a copiare indisturbato l'elemento A(r,c). La 136 rivela e stampa, invece, un elemento diagonale nullo creatosi durante le trasformazioni, segno evidente di determinante nullo nella matrice dei coefficienti.

Riportiamo una passata di giomin3 per dare un esempio di trattamento di una matrice e la sua riduzione a una forma equivalente diagonale. In tal modo appare evidente, riferendosi alla matrice dei coefficienti delle incognite in un sistema di equazioni di pari ordine, che la matrice diagonale derivata e' gia' il preambolo della soluzione del sistema. Infatti, in un sistema di equazioni che avesse una matrice dei coefficienti pari alla matrice diagonale ottenuta da quella dei coefficienti del sistema di partenza, siamo molto vicini alla soluzione del sistema di equazioni perche' in ogni equazione appare solo un'incognita. Non si puo' ancora dire, pero', che, per ottenere il valore di quell'incognita, basti dividere il termine noto al secondo membro per il coefficiente dell'incognita stessa.

```
CALCOLO DELLA MATRICE DIAGONALE   -PASSO A PASSO (s/n)   -DIMENS. MATRICE=3

  a( 1 , 1 )=1;  a( 1 , 2 )=-1;  a( 1 , 3 )=2;
  a( 2 , 1 )=1;  a( 2 , 2 )=-2;  a( 2 , 3 )=1;
  a( 3 , 1 )=1;  a( 3 , 2 )=1;  a( 3 , 3 )=2;

LA MATRICE APPENA INTRODOTTA
 1 -1  2
 1 -2  1
 1  1  2

 LA MATRICE DIAGONALE
 1  0  0
 0 -1  0
 0  0 -2
```

Vedremo nel paragrafo A.5 un metodo per risolvere un sistema di **n** equazioni in **n** incognite fondato sulla ricerca della matrice diagonale ottenuta da quella dei coefficienti del sistema.

A.1.4 IL CALCOLO DELLA MATRICE INVERSA

Ora che abbiamo visto qualche esempio di trattamento di matrici, vediamo come calcolare l'inversa di una matrice. La cosa riveste importanza non tanto per arricchire il repertorio del calcolo matriciale, quanto perche', come gia' osservato, un metodo per risolvere un sistema di **n** equazioni in **n** incognite consiste proprio nel calcolare

l'inversa della matrice dei coefficienti e moltiplicarla per il vettore colonna dei termini noti. Secondo la notazione vettoriale:

$$\text{Se} \quad \mathbf{Ax} = \mathbf{b} \quad , \quad \mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \; .$$

Il seguente programma, denominato **minv3**, serve appunto a calcolare l'inversa di una data matrice.

Il procedimento opera simultaneamente su una matrice composta dalla matrice da invertire e dalla matrice unitaria di uguale ordine; all'ultimo passaggio, quando la matrice data si e' trasformata in matrice unitaria, le stesse operazioni elementari hanno trasformato anche la matrice unitaria. L'assetto finale di quest'ultima e' la matrice inversa della matrice data. Per verificare il risultato basta moltiplicare la matrice di partenza e la sua inversa con il programma **molgen** il cui nucleo, in minv3 e' stato ripetuto dalla linea 370 in poi.

```
10 'Minv3
20 CLS:KEY OFF:PRINT"CALCOLO DELLA MATRICE INVERSA";
30 INPUT " *****  DIMENSIONE MAT.=",D
40 PRINT:FOR R=1 TO D:FOR C=1 TO D
50 PRINT "  f("R","C")=";:INPUT ;"",F(R,C)
60 F1(R,C)=F(R,C):NEXT:PRINT:NEXT
70 FOR R=1 TO D'......GENERAZIONE MATRICE UNITARIA
80 FOR C=1 TO D:IF R=C THEN U(R,C)=1 ELSE U(R,C)=0
90 NEXT:NEXT
100 FOR R=1 TO D-1
110 IF ABS(F(R,R))>.00001 THEN 140
120 PRINT "limite di prec.";F(R,R)
130 STOP
140 FOR C=R+1 TO D
150 M=F(C,R)/F(R,R)
160 FOR Q=1 TO D
170 F(C,Q)=F(C,Q)-M*F(R,Q)
180 U(C,Q)=U(C,Q)-M*U(R,Q)
190 NEXT:NEXT:NEXT:PRINT
200 PRINT"F(r,c) dopo le operazioni sulle righe"
210 FOR R=1 TO D:FOR C=1 TO D
220 PRINT F(R,C);:NEXT:PRINT:NEXT:PRINT
230 PRINT"U(r,c) dopo le operazioni sulle righe"
240 FOR R=1 TO D:FOR C=1 TO D
250 PRINT U(R,C);:NEXT:PRINT:NEXT:PRINT
260 PRINT "LA MATRICE INVERSA:"
270 FOR R=D TO 1 STEP -1
280 FOR Q=1 TO D
290 IF R=D THEN 330
300 FOR C=R+1 TO D
310 U(R,Q)=U(R,Q)-F(R,C)*U(C,Q)
320 NEXT C
330 U(R,Q)=U(R,Q)/F(R,R)
340 NEXT Q:NEXT R
350 FOR R=1 TO D:FOR C=1 TO D
360 PRINT U(R,C);:NEXT:PRINT:NEXT
```

```
370 'Calcolo di verifica:F1(r,c) x TU(r,c)
380 FOR C=1 TO D:FOR R=1 TO D:FOR Q=1 TO D
390 P(R,C)=P(R,C)+F1(R,Q)*U(Q,C):NEXT:NEXT:NEXT:PRINT
400 PRINT"LA MATRICE PRODOTTO:"
410 FOR R=1 TO D:FOR C=1 TO D
420 PRINT P(R,C);:NEXT:PRINT:NEXT:END
```

Dalla seguente passata si rileva che il programma, prima di stampare il risultato, visualizza le trasformazioni subite dalla matrice data e dalla matrice unitaria di pari ordine in seguito ai trattamenti operati sulle righe di entrambe. Si noti che la verifica, eseguita moltiplicando la matrice di partenza e la sua inversa, ha dato per risultato la matrice unitaria.

```
CALCOLO DELLA MATRICE INVERSA *****  DIMENSIONE MAT.=3

  f( 1 , 1 )=1  f( 1 , 2 )=-1  f( 1 , 3 )=2
  f( 2 , 1 )=1  f( 2 , 2 )=-2  f( 2 , 3 )=1
  f( 3 , 1 )=1  f( 3 , 2 )=1  f( 3 , 3 )=2

F(r,c) dopo le operazioni sulle righe
 1 -1  2
 0 -1 -1
 0  0 -2

U(r,c) dopo le operazioni sulle righe
 1  0  0
-1  1  0
-3  2  1

LA MATRICE INVERSA:
-2.5  2  1.5
-.5  0  .5
 1.5 -1 -.5

LA MATRICE PRODOTTO:
 1  0  0
 0  1  0
 0  0  1
```

A.1.5 RISOLUZIONE DI SISTEMI DI EQUAZIONI LINEARI

Quanto fin qui esaminato sul calcolo delle matrici ha trattato gli elementi propedeutici per avvicinarci ad alcuni metodi di risoluzione dei sistemi di equazioni lineari. In particolare, i programmi **giomin3** e **minv3** hanno consentito di sperimentare trattamenti diversi applicabili alle matrici quadrate. Il primo, giomin3, converte una matrice nella sua equivalente diagonale, mentre il secondo, opera in modo da calcolare l'inversa di una matrice data.

Sia dato quindi un sistema di **n** equazioni lineari in **n** incognite; un metodo di risoluzione discende immediatamente dalla definizione di inversa di una matrice. Infatti, se calcoliamo l'inversa della matrice dei coefficienti delle incognite di un sistema di equazioni e la moltiplichiamo per il vettore colonna **b** dei termini noti, otterremo un vettore colonna i cui elementi sono tanti quante sono le righe della matrice dei coefficienti e, nell'ordine, rappresentano, quindi, i valori delle incognite che soddisfano il sistema.

Operativamente, occorre procedere nel seguente modo. Si introducano gli elementi della matrice dei coefficienti nel programma **minv3** e, calcolata l'inversa, si utilizzi il programma **molgen** introducendo in esso, come matrici fattori, le seguenti: nell'ordine, prima gli elementi dell'inversa dei coefficienti e poi quelli del vettore colonna, cioe' il vettore dei termini noti. Il prodotto, che e' la soluzione del sistema dato, e' quanto eseguito nel seguente stampato.

```
F( 1 , 1 )=-2.5  F( 1 , 2 )=2   F( 1 , 3 )=1.5
F( 2 , 1 )=-.5  F( 2 , 2 )=   F( 2 , 3 )=.5
F( 3 , 1 )=1.5  F( 3 , 2 )=-1  F( 3 , 3 )=-.5

G( 1 , 1 )=-1
G( 2 , 1 )=2
G( 3 , 1 )=1

-2.5  2  1.5
-.5  0  .5
 1.5 -1 -.5

-1
 2
 1

LA MATRICE PRODOTTO:

 8
 1
-4
```

L'altro metodo di risoluzione si aggancia all'algoritmo che sta nel nucleo di giomin3 e che consiste nel trattare, con operazioni elementari fino alla diagonalizzazione, una matrice composta dalla matrice dei coefficienti insieme al vettore colonna dei termini noti: cosi', una volta ridotta la matrice dei coefficienti ai soli elementi sulla diagonale, anche i termini noti saranno stati sottoposti alle stesse operazioni e il risultato sara' un sistema veramente equivalente al sistema dato, ma con una sola incognita in ciascuna equazione. Al limite, se la matrice diagonale fosse anche unitaria, il vettore colonna dei termini noti dopo il trattamento costituirebbe anche il vettore soluzione del sistema. Altrimenti, basta dividere il termine noto ridotto per l'elemento corrispondente sulla diagonale dei coefficienti. Un programma che fa tutto questo e' **jovar3** di cui riportiamo il listato e alcune passate; di queste la prima si riferisce all'esempio illustrato nella figura A.4.

```
10 'jovar3
20 CLS:KEY OFF:PRINT"RISOLUZIONE DI UN SISTEMA DI EQUAZIONI";
25 PRINT" LINEARI   ---";CHR$(16);:INPUT "N.EQUAZIONI=",RT$
30 RT=VAL(RT$):IF RT=0 THEN 20
50 FOR R=1 TO RT:FOR C=1 TO RT+1
68 PRINT "a("R","C")=";:INPUT; "",AS(R,C):PRINT"; ";
70 NEXT:PRINT:NEXT:PRINT
80 FOR R=1 TO RT
90 FOR C=1 TO RT+1
100 PRINT AS(R,C);
120 NEXT:PRINT:NEXT
1000 FOR I=1 TO RT  'salvataggio dei coefficienti
1020 FOR J=1 TO RT+1:A(I,J)=AS(I,J):NEXT J:NEXT I
2000 'modved: per vedere e modificare i coefficienti
2010 PRINT:PRINT"Vuoi vedere o modificare qualche "FF$" dato ? (s/n)  ";
2020 DD$=INPUT$(1):PRINT DD$;:IF DD$="s" THEN GOSUB 20000 ELSE GOTO 5000
2040 GOTO 2020
2060 DS$=INPUT$(1):IF DS$<>"v" AND DS$<>"f" AND DS$<>"l" THEN 2040
2080 IF DS$="l" THEN SC=1:IF DS$="f" THEN SC=2:STOP:IF DS$="v" THEN SC=3
2100 IF DD$="s" THEN GOSUB 20000 ELSE GOTO 5000
2120 GOTO 2020 'XXX.V.XII
5000 'cal+stam
5010 PRINT:PRINT:FOR R=1 TO RT
5015 FOR C=1 TO RT+1
5017 PRINT A(R,C);
5018 NEXT:PRINT:NEXT:PRINT STRING$(50,95)
5020 FOR Q=1 TO RT
5040 FOR R=1 TO RT
5060 IF A(Q,Q)=0 THEN PRINT "A("Q","Q")=0":END
5080 M=-A(R,Q)/A(Q,Q)
5100 FOR C=1 TO RT+1
5120 IF Q=R THEN M=0:GOTO 5140
5140 A(R,C)=M*A(Q,C)+A(R,C)
5240 NEXT:NEXT:NEXT:PRINT
5320 'END
10000 'stampa matrice
10040 FOR S=1 TO RT:FOR T=1 TO RT+1
10060 Z=A(S,T)
10080 IF Z=0 THEN 10120
10100 IF ABS(Z)<1E-09 THEN A(S,T)=0
10120 PRINT A(S,T);
10140 NEXT:PRINT:NEXT:PRINT
10200 FOR S=1 TO RT
10220 Y=A(S,RT+1)/A(S,S)
10240 PRINT "A("S","RT+1")="Y
10260 NEXT
10280 GOTO 1000
20000 'rued (routine di editing):per vedere o modificare i coefficienti
20020 PRINT:PRINT"VEDERE O MODIFICARE (m/v) ? ";
20040 QQ$=INPUT$(1):PRINT " "QQ$;
20050 IF QQ$="m" THEN  R$="modificare" ELSE R$="vedere"
20060 PRINT:PRINT:LOCATE 22,1:PRINT"Quale vuoi "R$" ?"
20080 PRINT "(T)ermine noto - (C)oefficiente (T/C)"
```

```
20100 PRINT STRING$(50,32)
20120 DS$=INPUT$(1)
20140 IF DS$="t" THEN SC=1
20160 IF DS$="c" THEN SC=2
20200 IF QQ$="m" THEN 20260
20220 IF QQ$="v" THEN 20680
20240 IF QQ$<>"m" OR QQ$<>"v" THEN PRINT" ###":RETURN
20260 'modificare
20280 ON SC GOTO 20300,20520
20300 INPUT "termine noto =",R
20320 PRINT "T("R")=";
20340 INPUT; " ",MP:PRINT
20360 A(R,RT+1)=MP
20380 GOTO 20640
20400 '
20420 INPUT "n. coeff. F.O.=",C
20440 PRINT "cf("C")=";
20460 INPUT; " ",MC:PRINT
20480 CS(C)=MC:C(C)=MC
20500 GOTO 20640
20520 '
20540 INPUT "n. riga = ",R
20560 INPUT "n. col. = ",C
20580 PRINT "CV("R","C")=";
20600 INPUT; " ",MV:PRINT
20620 AS(R,C)=MV:A(R,C)=MV
20640 '
20660 FF$="altro":RETURN
20680 'vedere
20700 ON SC GOTO 20720,20840
20720 INPUT "termine noto = ",R
20740 PRINT "T("R")="A(R,RT+1);
20760 GOTO 20900
20780 INPUT "n. coefficiente = ",F
20800 PRINT "cf("C")="C(C);
20820 GOTO 20900
20830 '----->::::::>>>
20840 INPUT "n.riga coeff.=",R
20860 INPUT "n.colonna coeff.=",C
20880 PRINT "CV("R","C")="A(R,C);
20900 FF$="altro":RETURN
Ok
RUN
RISOLUZIONE DI UN SISTEMA DI EQUAZIONI LINEARI   ---N.EQUAZIONI=3
a( 1 , 1 )=2; a( 1 , 2 )=2; a( 1 , 3 )=2; a( 1 , 4 )=1;
a( 2 , 1 )=1; a( 2 , 2 )=-1; a( 2 , 3 )=1; a( 2 , 4 )=1;
a( 3 , 1 )=0; a( 3 , 2 )=0; a( 3 , 3 )=2; a( 3 , 4 )=1;

 2  2  2  1
 1 -1  1  1
 0  0  2  1

Vuoi vedere o modificare qualche  dato ? (s/n)  n
```

```
Vuoi vedere o modificare qualche  dato ? (s/n)  n

 2  2  2  1
 1 -1  1  1
 0  0  2  1
__________________________________________________

 2  0  0  .5
 0 -2  0  .5
 0  0  2  1

A( 1 , 4 )= .25
A( 2 , 4 )=-.25
A( 3 , 4 )= .5

Vuoi vedere o modificare qualche altro dato ? (s/n)  s
VEDERE O MODIFICARE (m/v) ?  m

Quale vuoi modificare ?
(T)ermine noto - (C)oefficiente (T/C)

n. riga = 3
n. col. = 3
CV( 3 , 3 )= 1


 2  2  2  1
 1 -1  1  1
 0  0  1  1
__________________________________________________

 2  0  0 -.5
 0 -2  0  .5
 0  0  1  1

A( 1 , 4 )=-.25
A( 2 , 4 )=-.25
A( 3 , 4 )= 1

Vuoi vedere o modificare qualche altro dato ? (s/n)
```

Si noti che, una volta raggiunto il risultato,il programma accetta variazione su uno o piu' elementi del sistema, senza dover reintrodurre quelli invariati.

Cio' consente, da un lato di correggere eventuali errori di introduzione e, dall'altro, di sperimentare la sensibilita' di un fattore sul vettore soluzione.

Nel caso sopra riportato, per esempio, si e' prima modificato un coefficiente e, successivamente, un termine noto.

A.2 ELEMENTI DI PROGRAMMAZIONE LINEARE

Per programmazione lineare (LP) s'intende un metodo di ricerca del valore ottimo (se esiste) di una funzione lineare di piu' variabili in un campo limitato di esistenza di esse. Il termine **programmazione**, quindi, non ha niente a che vedere con i calcolatori elettronici, riguardando, invece, una tecnica di programmazione dell'utilizzo di risorse in problemi complessi, in cui la scelta migliore, o comunque quella piu' conveniente, e' difficilmente determinabile.

I calcolatori elettronici, pero', giocano un ruolo importante nel trattare questo ramo della Ricerca Operativa, da quando, nel 1947, G.B. Dantzing mise a punto una procedura di calcolo (il metodo del simplesso) che puo' essere applicata a problemi di grandi dimensioni solo con l'ausilio di mezzi informatici.

A.2.1 UN ESEMPIO CLASSICO CHIARIFICATORE

Per introdurre il lettore nell'argomento prendiamo uno degli esempi piu' diffusi: quello di un programma di produzione che prevede due fattori produttivi, l'energia e la materia prima. Entrambi i fattori hanno un costo che entra in misura diversa nel costo di produzione di due articoli il cui prezzo e i volumi di vendita influiscono sul profitto netto complessivo. Per formalizzare il problema e tentarne la soluzione dovremo fissare alcune ipotesi semplificative sufficientemente plausibili. Cominciamo dalla terminologia per poterci esprimere chiaramente e in modo sintetico.

Siano x_1 e x_2 le quantita' da produrre dei due articoli e c_1, c_2 i ricavi unitari, ovvero prezzi, dalla vendita degli stessi. Inoltre, le disponibilita' dei due fattori produttivi siano limitate ai valori E e M, indicanti, rispettivamente, i limiti massimi dell'energia elettrica e della materia prima. Tali limiti vanno messi in relazione con le quantita' prodotte dell'uno e dell'altro articolo, tenuto conto che, generalmente, anche la produzione complessiva non deve superare un dato valore P.
Si abbiano, ad esempio, i seguenti parametri tecnici:

a_{11} la quantita' di M per produrre 1 tonnellata dell'articolo 1

a_{12} la quantita' di M per produrre 1 tonnellata dell'articolo 2

a_{21} la quantita' di E per produrre 1 tonnellata dell'articolo 1

a_{22} la quantita' di E per produrre 1 tonnellata dell'articolo 2.

In regime di proporzionalita' tra fattori produttivi e quantita' prodotte, potremo scrivere:

$a_{11}x_1 + a_{12}x_2 \leq M$ vincolo della materia prima

$a_{21}x_1 + a_{22}x_2 \leq E$ vincolo dell'energia

$a_{31}x_1 + a_{32}x_2 \leq P$ vincolo della produzione massima

Resta da esprimere il criterio in base al quale scegliere le quantita' da produrre per realizzare un certo obiettivo. Tale criterio, nell'esempio fatto, viene formalizzato legando le quantita' incognite x_1 e x_2 ai ricavi netti:

$$z = c_1x_1 + c_2x_2$$

Trattandosi di coefficienti di ricavo, o prezzi, il problema da risolvere e' quello di cercare i valori di x_1 e x_2 che, nel rispetto delle relazioni vincolari, rendono massima la funzione **z**. Dalla linearita' di tali relazioni e da quella della funzione obiettivo **z** discende che il valore ottimo sara' raggiunto su un vertice della regione ammissibile definita dai vincoli. Il metodo del simplesso non solo si avvale di questa circostanza, ma dispone inoltre di un metodo di ricerca dell'estremo che semplifica notevolmente il calcolo. Iniziando, infatti, il calcolo da un qualsiasi vertice della regione ammissibile, con un particolare algoritmo, si passa a un vertice adiacente scelto tra tutti i vertici adiacenti tale che il valore della funzione obiettivo risulti aumentato: si dimostra che tale vertice esiste sempre, a meno che l'ottimo non sia gia' stato raggiunto.

Assegniamo ora valori numerici ai vincoli, ai coefficienti tecnici e a quelli della funzione obiettivo e diamo al problema un'interpretazione geometrica. La funzione da massimizzare sia

$$z = 100x_1 + 50x_2 \quad .$$

Essa rappresenta un piano passante per l'origine degli assi, avente come intercette:

col piano $z=0$	la retta	$100x_1 + 50x_2 = 0$;
col piano $x_2=0$	la retta	$z = 100x_1$;
col piano $x_1=0$	la retta	$z = 50x_2$.

La funzione **z** e' sottoposta alle relazioni vincolari, generalmente in numero NL<NV (per cui si hanno NL equazioni in NV incognite). In tal caso il problema e' indeterminato in quanto esistono infiniti valori delle incognite che soddisfano le equazioni: si tratta di scegliere quei valori non negativi che rendono ottima la **z**.

Nel caso in esame, si vogliono ricercare quelle quantita' non negative che rendono massima la funzione, nel rispetto delle seguenti relazioni:

1)	$10x_1 + 20x_2 \leq 1000$	AB' : (0,50); (100,0)
2)	$300x_1 + 100x_2 \leq 12000$	F'D : (0,120); (40,0)
3)	$x_1 + x_2 \leq 60$	B'C': (0,60); (60,0)

Per la rappresentazione geometrica delle relazioni vincolari (figura A.5) ricordiamo che esse, come disequazioni, rappresentano una porzione del piano x_1x_2.

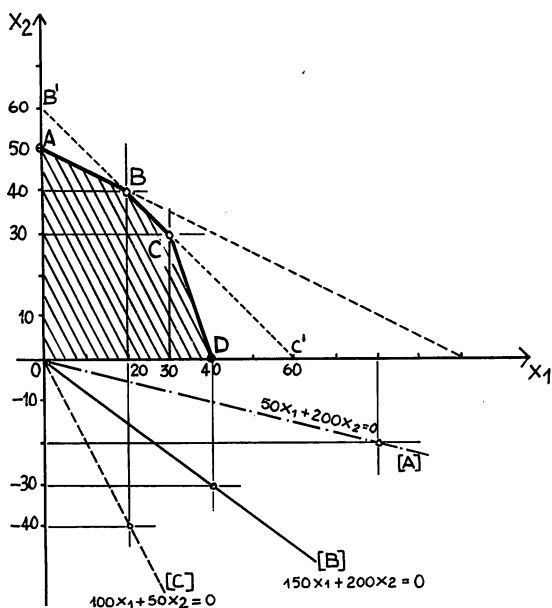



Figura A.5

Pero', raggiungendo il limite (segno uguale), esse rappresentano delle rette di separazione di due porzioni del piano. Nel problema in esame, poiche' nella disequazione c'e' il segno [<], la porzione interessata e' quella sottostante. Le intercette di tali rette con gli assi cartesiani hanno rispettivamente, i valori riportati a fianco delle disequazioni prima riportate. E' importante notare come, nei problemi con due variabili di controllo, sia possibile arrivare per via grafica alla soluzione nella ricerca del massimo della funzione obiettivo. Basta tracciare delle rette parallele a quella, passante per l'origine, per cui z=0 (la retta intersezione tra il piano z=0 e la funzione obiettivo). Il valore di z sara' tanto maggiore quanto piu' e' distante la retta dall'origine. L'ottimo si ottiene quando la retta parallela e' alla massima distanza dall'origine, pur avendo almeno un punto in comune col contorno della regione delle soluzioni ammesse. Tale punto, se esiste, si dimostra essere in un uno dei vertici della poligonale individuata dai vincoli. In tal modo, nella figura A.5, abbiamo trovato il vertice C(30,30) in cui la funzione $z = 100x_1 + 50x_2$ e' massima.

A parita' di relazioni vincolari, si noti, il vertice puo' non essere lo stesso, come appare per gli altri due tipi di funzioni obiettivo rappresentate [A] e [B]. Nello spazio (figura A.6), le tre relazioni vincolari sono le intersezioni di tre piani paralleli all'asse z col piano z=0. Gli spigoli sono anch'essi rette parallele all'asse z, che intersecano il piano della funzione obiettivo $z=100x_1 + 50x_2$ nei punti A", B", C", D" di cui vediamo le proiezioni A, B, C, D sul piano x_1,x_2.

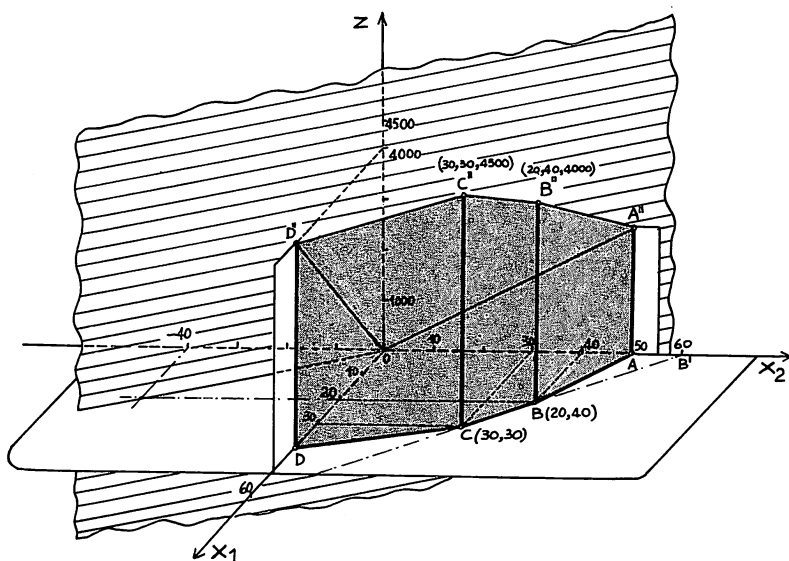



Figura A.6

Tra i vertici A", B", C", D" quello che corrisponde al massimo della funzione e' il vertice C" di coordinate (30,30,4500): 4500 e' appunto il valore massimo della funzione obiettivo, ottenuto per i valori x_1=30, x_2=30 delle quantita' da produrre dei due articoli.

Nei problemi di programmazione lineare piu' complessi, ovviamente, non si puo' far ricorso alla risoluzione grafica ora descritta, ma le definizioni e i metodi continuano a valere. In generale, si puo' dimostrare che, giunti alla soluzione di un problema di ottimo in cui vi siano NL relazioni vincolari e NV incognite, almeno NV-NL variabili devono essere nulle. Come generalizzazione del caso esaminato all'inizio, in cui le variabili erano solo due, ora, se le variabili sono in numero di NV, la soluzione si trova in corrispondenza di un vertice dell'iperpoliedro convesso; tale vertice estremante e' uno tra tutti i vertici delle possibili soluzioni, che il calcolo combinatorio indica essere in numero di:

$$n = \frac{NV!}{(NV-NL)!\ NL!} .$$

Si ricordi che per a! (fattoriale di a) si intende il prodotto dei primi a numeri interi. (Convenzionalmente 0!=1.) La formula indicata, che fornisce come limite il numero delle soluzioni di base possibili, e' il numero delle combinazioni di NV oggetti, di classe NL. Ad esempio per NV=7 e NL=4, il numero delle combinazioni di 7 variabili, prese a quattro a quattro, e' n = 35.
In questo caso potremmo anche calcolare tutti i valori della funzione

obiettivo e selezionare il migliore; ma questa strada sarebbe impraticabile per valori di NV e NL superiori; infatti gia' per NV=20 e NL=10 il numero delle combinazioni sale a 184756. In tali casi l'impiego di un algoritmo particolare come il simplesso e di un calcolatore costituiscono l'unico modo di affrontare il problema.

A.2.2 IL SIMPLESSO

Il metodo del simplesso e' un algoritmo mediante il quale, partendo da una soluzione di partenza (detta di **base**), si calcola la funzione da ottimizzare. Dalla soluzione di base iniziale, successivamente, si elimina una variabile e al suo posto ne entra un'altra; la nuova situazione viene ancora valutata in termini di funzione obiettivo per accertare se questa ha subito un miglioramento. La procedura si ripete finche' la funzione non migliora piu'. Per l'applicazione del metodo del simplesso occorre:

- definire l'obiettivo da massimizzare o minimizzare ed esprimerlo in forma lineare rispetto alle variabili di controllo;
- stabilire le relazioni vincolari cui la funzione obiettivo e' sottoposta ed esprimerle in forma lineare rispetto alle variabili.

Quando si definisce la forma dell'obiettivo si e' soliti esprimerla come funzione da massimizzare; cosi' se si deve trovare il minimo di una funzione

$$z = c_1x_1 + c_2x_2 \quad ,$$

per convertirla in una funzione da massimizzare, e' sufficiente cercare il massimo della funzione

$$-z = -c_1x_1 - c_2x_2 \quad .$$

Occorre inoltre trovare un artificio per trattare le disuguaglianze eventualmente presenti nelle relazioni vincolari; tale artificio consiste nel trasformare le disequazioni in equazioni, attraverso l'introduzione di variabili fittizie che avranno peso zero nella funzione obiettivo. Ad esempio:

(1) il vincolo $x_1 + x_2 \leq 60$ si trasforma in

$$x_1 + x_2 + x_3 = 60 \quad , \quad \text{con } x_3 \geq 0 \quad ;$$

(2) il vincolo $x_1 + x_2 \geq 60$ si trasforma in

$$x_1 + x_2 - x_3 = 60.$$

Anche nel caso in cui la relazione vincolare sia gia' un'equazione, occorre introdurre una variabile (detta di comodo) che, nella funzione obiettivo, avra' un coefficiente negativo molto alto.

A.2.3 IL PROGRAMMA DEL SIMPLESSO

Forniamo ora alcuni esempi di problemi di programmazione lineare al solo scopo di illustrare il funzionamento del programma **simp86**. Tale programma, in apertura, propone due tipi di presentazione di risultati: quella in cui si visualizza solo la tabella finale dalla quale risultano il valore ottimo raggiunto per la funzione obiettivo e i valori estremanti delle variabili di controllo e quella in cui si possono seguire passo a passo tutti i miglioramenti subiti dalla funzione obiettivo, durante la ricerca dell'ottimo. Per quanto riguarda l'introduzione dei dati occorre digitare i coefficienti della funzione obiettivo, i limiti e i coefficienti vincolari in ciascuna delle relazioni vincolari.

Prima di iniziare l'esecuzione dei calcoli, simp86 chiede se si vuole esaminare o modificare qualche dato gia' introdotto. La stessa domanda viene posta, quando si raggiunge il risultato. E' interessante sperimentare, per lo stesso problema, diversi valori di un parametro (ad esempio un coefficiente tecnico) a parita' di tutti gli altri, e notare l'influenza della variazione di un parametro sui valori estremanti delle variabili di controllo. In tal modo, si riescono a studiare le sensibilita' dei parametri nei confronti dell'ottimo.

Vediamo con l'esempio gia' analizzato per l'interpretazione geometrica il modo per normalizzare i dati da introdurre. Poiche' si tratta di un problema di massimo, la funzione da ottimizzare e' gia' nella forma canonica; per quanto riguarda i vincoli, trattandosi di disuguaglianze col segno di [<], occorre introdurre nella funzione obiettivo, dopo i termini relativi ai coefficienti delle variabili di controllo, i termini corrispondenti alle variabili fittizie. La funzione obiettivo sara' quindi formata di cinque termini: i primi due associati alle variabili di controllo, mentre gli ultimi tre hanno peso zero. Riportiamo una passata del programma per mostrare i dati introdotti e le soluzioni calcolate passo a passo.

```
SOLUZIONI INTERMEDIE (s/n)  SI
N.REL.VIN.=3
N.VAR.TOT.=5

I COEFFICIENTI DELLA F. OBIETTIVO:

C( 1 )= 100;  C( 2 )= 50;  C( 3 )= 0;  C( 4 )= 0;  C( 5 )= 0;

 I LIMITI VINCOLARI:

T( 1 )=1000;  T( 2 )=12000;  T( 3 )=60;

I COEFFICIENTI VINCOLARI:

A( 1 , 1 )=10; A( 1 , 2 )=20;
A( 2 , 1 )=300; A( 2 , 2 )=100;
A( 3 , 1 )=1; A( 3 , 2 )=1;

Vuoi vedere o modificare qualche  dato ? (s/n)
```

```
>-----> RISULTATI DEL PASSO ------> N. 1 :
C. F. OBB. 100   50   0   0   0
variabili  1   2   3   4   5
  3   1000   10   20   1   0   0
  4   12000   300   100   0   1   0
  5   60   1   1   0   0   1

F.OB.  0   100   50   0   0   0
per continuare BATTI (CR)
>-----> RISULTATI DEL PASSO ------> N. 2 :
C. F. OBB. 100   50   0   0   0
variabili  1   2   3   4   5
  3   600   0   16.66667   1   -3.333334E-02   0
  1   40   1   .3333334   0   3.333334E-03   0
  5   20   0   .6666666   0   -3.333334E-03   1

F.OB.  4000   0   16.66666   0   -.3333334   0
per continuare BATTI (CR)
>-----> RISULTATI DEL PASSO ------> N. 3 :
C. F. OBB. 100   50   0   0   0
variabili  1   2   3   4   5
  3   100   0   0   1   .05   -25
  1   30   1   0   0   5.000001E-03   -.5000001
  2   30   0   1   0   -5.000001E-03   1.5

F.OB.  4500   0   0   0   -.25   -25
```

Seguono i prospetti relativi agli obiettivi ottenuti con coefficienti diversi nella funzione obiettivo: si hanno, quindi, due nuove inclinazioni della retta obiettivo sul piano z=0 e, corrispondentemente, due punti diversi di tangenza al contorno (A e B) (figura A.5).

```
>-----> RISULTATI DEL PASSO ------> N. 2 :
C. F. OBB. 50   200   0   0   0
variabili  1   2   3   4   5
  2   50   .5   1   .05   0   0
  4   7000   250   0   -5   1   0
  5   10   .5   0   -.05   0   1

F.OB.  10000  -50   0  -10   0   0
I CALCOLI SONO TERMINATI. Tutto daccapo? (s/n)

>-----> RISULTATI DEL PASSO ------> N. 3 :
C. F. OBB. 150   200   0   0   0
variabili  1   2   3   4   5
  2   40   0   1   .1   0   -1
  4   2000   0   0   20   1   -500
  1   20   1   0   -.1   0   2

F.OB.  11000   0   0   -5   0   -100
I CALCOLI SONO TERMINATI. Tutto daccapo? (s/n)
```

Segue il programma **simp86**.

```
10 'simp86:1.1.86. APPLICAZIONI DI PROGRAMMAZIONE LINEARE. IL SIMPLESSO <-----
20 CLS:CLEAR:KEY OFF
30 PIT=0
40 LP=.0000002  'limite di precisione
50 PRINT "SOLUZIONI INTERMEDIE (s/n)";
60 WW$=INPUT $(1):IF WW$="s" THEN PRINT "  SI":T=1:GOTO 70
65 PRINT "  NO"
70 INPUT "N.REL.VIN.=",NL 'numero limiti vincolari
80 INPUT "N.VAR.TOT.=",NV 'numero variabili totale (effettive+fittizie)
90 FF$=""
100 DIM H(NV+NL),A(NL,NV),P(NL),CV(NV),C(NV+NL),RV(NV),CC(NV),CR(NV),HC(NV),CX(N
V)
110 DIM AS(NL,NV),PS(NL),CS(NV+NL)
120 FOR P=1 TO NV
130 IF P<=NL THEN 150
140 CV(P)=P-NL:GOTO 160
150 CV(P)=P-NL+NV
160 NEXT:PRINT
170 'PRINT
180 PRINT "I COEFFICIENTI DELLA F. OBIETTIVO:"
190 '*****
200 PRINT : FOR C=1 TO NV
210 PRINT "C("C")=";
220 CC=C+NL
230 IF CC>NV THEN CC=CC-NV
240 INPUT;" ",CS(CC):PRINT ";  ";'---->>>**********
250 NEXT C:PRINT
260 W=NL+1
270 'FOR I=1 TO NV:PRINT CS(I);:NEXT :STOP
280 PRINT
290 PRINT" I LIMITI VINCOLARI:":PRINT
300 FOR R=1 TO NL
310 PRINT "T("R")=";
320 INPUT ;"",PS(R):PRINT ";  ";
330 NEXT
340 PRINT:PRINT:PRINT"I COEFFICIENTI VINCOLARI:":PRINT
350 FOR R=1 TO NL
360 FOR C=W TO NV
370 PRINT "A("R","C-NL")=";
380 INPUT;"",AS(R,C):PRINT"; ";
390 NEXT C
400 PRINT
410 NEXT R
1000 FOR R=1 TO NL
1010 FOR C=1 TO NV:A(R,C)=AS(R,C):NEXT C:P(R)=PS(R):NEXT R
1020 FOR C=1 TO NV:C(C)=CS(C):NEXT:PRINT
1030 PIT=0:PRINT"Vuoi vedere o modificare qualche "FF$" dato ? (s/n)  ";:DD$=IN
UT$(1):IF DD$="s" THEN GOSUB 9000 ELSE IF DD$="n" GOTO 1080 ELSE LOCATE 24,1:PI
NT STRING$(79,32);:GOTO 1030
1040 GOTO 1030
1050 DS$=INPUT$(1):IF DS$<>"v" AND DS$<>"f" AND DS$<>"1" THEN 1040
1060 IF DS$="1" THEN SC=1:IF DS$="f" THEN SC=2:IF DS$="v" THEN SC=3
1070 IF DD$="s" THEN GOSUB 9000 ELSE GOTO 1080
```

```
1080 FOR R=1 TO NL
1090 FOR C=1 TO NL
1100 A(R,C)=0
1110 NEXT C
1120 CR(R)=C(R)
1130 RV(R)=CV(R)
1140 A(R,R)=1
1150 NEXT R
1160 LP=.0000002:UN=-7.978499E+20
1170 FOR C=1 TO NV
1180 H=0
1190 FOR R=1 TO NL
1200 H=H+A(R,C)*CR(R)
1210 NEXT R
1220 BQ=C(C)-H
1230 HC(C)=H
1240 CX(C)=BQ
1250 IF UN>BQ THEN 1280
1260 UN=BQ
1270 CR=C
1280 NEXT C
1290 PIT=PIT+1
1300 IF T<>0 THEN 5000
1310 IF UN>=LP THEN 7000
5000 OB=0
5010 FOR R=1 TO NL
5020 OB=OB+P(R)*CR(R)
5030 NEXT R
5040 'PRINT
5050 PRINT ">-----> RISULTATI DEL PASSO ------> N."PIT":"
5060 PRINT "C. F. OBB.";
5070 FOR C=1 TO NV
5080 CC=C+NL
5090 IF CC>NV THEN CC=CC-NV
5100 PRINT C(CC)" ";
5110 NEXT C
5120 PRINT
5130 PRINT "variabili";
5140 FOR C=1 TO NV
5150 CC=C+NL
5160 IF CC>NV THEN CC=CC-NV
5170 PRINT " "CV(CC);
5180 NEXT C
5190 PRINT
5200 FOR R=1 TO NL
5210 PRINT " ";RV(R);
5220 PRINT " ";P(R);
5230 FOR C=1 TO NV
5240 CC=C+NL
5250 IF CC>NV THEN CC=CC-NV
5260 PRINT " "A(R,CC);
5270 NEXT C
5280 PRINT
```

```
5290 NEXT R
5300 PRINT
5310 PRINT "F.OB.";" " ;OB;
5320 FOR C=1 TO NV
5330 CC=C+NL
5340 IF CC>NV THEN CC=CC-NV
5350 PRINT " ";CX(CC);
5360 NEXT C
5370 PRINT
5380 IF UN<LP THEN GOTO 7295
5390 INPUT "per continuare BATTI (CR)",QQ$
7000 BB=1D+22
7010 FOR R=1 TO NL
7020 IF A(R,CR)<=LP THEN 7070
7030 DX=P(R)/A(R,CR)
7040 IF BB<DX THEN 7070
7050 BB=DX
7060 RX=R
7070 NEXT R
7080 IF BB<900000000000000# THEN 7110
7090 PRINT " la funzione obb. e' infinita"
7100 GOTO 7300
7110 PX=A(RX,CR)
7120 P(RX)=P(RX)/PX
7130 FOR C=1 TO NV
7140 A(RX,C)=A(RX,C)/PX
7150 NEXT C
7160 FOR R=1 TO NL
7170 IF R=RX THEN 7240
7180 P(R)=P(R)-P(RX)*A(R,CR)
7190 FOR C=1 TO NV
7200 IF C=CR THEN 7220
7210 A(R,C)=A(R,C)-A(R,CR)*A(RX,C)
7220 NEXT C
7230 A(R,CR)=0
7240 NEXT R
7250 LP=LP+.0000002
7260 CR(RX)=C(CR)
7270 RV(RX)=CV(CR)
7280 A(RX,CR)=1
7290 SOUND 2500,.5:GOTO 1160
7295 FOR Y=1 TO 9:FOR LUN=37 TO 4000 STEP 300:SOUND LUN,.3:NEXT:NEXT
7300 PRINT "I CALCOLI SONO TERMINATI. Tutto daccapo? (s/n)";:Q$=INPUT$(1)
7310 IF Q$="s" THEN 20 ELSE GOTO 1000
9000 PRINT:PRINT"VEDERE O MODIFICARE (m/v) ?"
9010 QQ$=INPUT$(1)
9020 PRINT:LOCATE 22,1:PRINT"Quale ? ---> ";
9030 PRINT "(L)imite Vinc.- coeff.(F).Obiett.- coeff.(V)inc. (L/F/V)"
9040 PRINT STRING$(50,32)
9050 DS$=INPUT$(1)
9060 IF DS$="l" THEN SC=1
9070 IF DS$="f" THEN SC=2
9080 IF DS$="v" THEN SC=3
```

```
9090 IF QQ$="m" THEN 9120
9100 IF QQ$="v" THEN 9350
9110 IF QQ$<>"m" OR QQ$<>"v" THEN PRINT" ###":RETURN
9120 PRINT"MOD."'-----------------------
9130 ON SC GOTO 9140,9190,9260
9140 INPUT "n. vincolo=",R
9150 PRINT "T("R")=";
9160 INPUT; " ",MP:PRINT
9170 PS(R)=MP:P(R)=MP
9180 GOTO 9330
9190 '
9200 INPUT "n. coeff. F.O.=",CX
9210 IF CX<=NL THEN C=CX+NV-NL ELSE C=CX+NL-NV
9220 PRINT "cf("CX")=";
9230 INPUT; " ",MC:PRINT
9240 CS(C)=MC:C(C)=MC
9250 GOTO 9330
9260 '
9270 INPUT "n. vincolo (riga)=",R
9280 INPUT "n. vincolo (col.)=",CX
9290 IF CX<=NL THEN C=CX+NV-NL ELSE C=CX+NL-NV
9300 PRINT "CV("R","CX")=";
9310 INPUT; " ",MV:PRINT
9320 AS(R,C)=MV:A(R,C)=MV
9330 PIT=0:LP=.0000005:UN=-8.999999E+19
9340 FF$="altro":RETURN
9350 PRINT "VED"
9360 ON SC GOTO 9370,9410,9460
9370 LOCATE 24,1:INPUT; "n. vincolo=",R
9380 IF R>NL OR R=0 THEN LOCATE 24,1:PRINT STRING$(40,32);:GOTO 9370
9390 PRINT "   T("R")="P(R)
9400 GOTO 9520
9410 INPUT "n.coeff. F.O.=",CX
9420 IF CX>NV THEN 9410
9430 IF CX<=NL THEN C=CX+NV-NL ELSE C=CX+NL-NV
9440 PRINT "cf("CX")="C(C);
9450 GOTO 9520
9460 INPUT "n.riga coeff. vinc.=",R
9470 IF R>NL THEN 9460
9480 INPUT "n.colonna coeff. vin.=",CX
9490 IF CX>NV THEN 9480
9500 IF CX<=NL THEN C=CX+NV-NL ELSE C=CX+NL-NV
9510 PRINT "CV("R","CX")="A(R,C);
9520 FF$="altro":RETURN
```

Come si puo' vedere dal listato, il programma e' stato protetto da introduzioni errate in misura sufficiente per un attento operatore; ma non si e' voluto appesantirlo eccessivamente per consentirne una piu' semplice leggibilita'. Il lettore esigente puo' partire da questa base e farne una propria versione, con le sue personali routine di introduzione; altrettanto dicasi per la presentazione dei risultati, ottenibili sicuramente in forma piu' elegante.

IL PROBLEMA DEI TRASPORTI

Alla classe dei problemi richiedenti la minimizzazione dei costi di distribuzione tra origine e destinazione di merci appartiene non solo quello classico dei trasporti, ma molti altri problemi di tipo industriale, come, per citarne uno, quello dell'assegnazione di risorse a parti o settori di impianto, assicurando le quantita' sufficienti al loro funzionamento e nello stesso tempo sfruttando al massimo le risorse impegnate globali. Qui trattiamo il classico problema del trasporto di merci che prodotte in date quantita' in determinate fabbriche devono essere dislocate in magazzini periferici per l'opportuna distribuzione sui relativi mercati. L'esempio numerico trattato e' semplicissimo, ma propedeutico per la particolare natura mista dei vincoli che sono in parte delle equazioni e in parte disequazioni.

Immaginiamo tre fabbriche localizzate a Milano, Genova e Bari che producono rispettivamente 100, 90 e 80 tonnellate della stessa merce da distribuire ai quattro magazzini periferici di Firenze, Roma, Napoli e Palermo, che, per alimentare i rispettivi mercati di vendita, richiedono **almeno** 100, 90, 30 e 50 tonnellate alla settimana. E' bene, innanzitutto, compilare una tabella dei costi di trasporto della merce che, nella fattispecie, supponiamo sia la seguente:

	MAGAZZINI				si produce non piu' di
FABBRICHE	1 FI	2 RM	3 NA	4 PA	
1 MI	4	8	10	16	100
2 GE	5	7	8	15	90
3 BA	12	10	8	11	80
almeno	100	90	30	50	270

La funzione obiettivo e' costituta, nell'ipotesi di linearita', dalla seguente sommatoria:

$$F_o = c_{11}x_1 + c_{12}x_2 + c_{13}x_3 + c_{14}x_4 + c_{21}x_5 + c_{22}x_6 + c_{23}x_7 + c_{24}x_8 +$$
$$+ c_{31}x_9 + c_{32}x_{10} + c_{33}x_{11} + c_{34}x_{12}.$$

Dall'esame delle relazioni vincolari, riprodotte per semplicita' nella tabella di figura A.7, risultano tre vincoli con segno di [≤] e quattro con segno [=]; di conseguenza per i primi tre si generano delle variabili fittizie di compensazione che avranno peso zero nella funzione obiettivo, mentre gli ultimi quattro , a loro volta, generano 4 variabili di comodo che avranno un peso negativo molto grande.

Normalizzando i vari aspetti, si giunge alla vera funzione obiettivo da massimizzare contenente tutte le variabili, effettive e di servizio, come risulta dal prospetto elaborato dal calcolatore.

Si noti che dovendo minimizzare un costo complessivo di trasporto, la funzione obiettivo deve subire una normalizzazione alla forma canonica di ricerca di un massimo e pertanto tutti i suoi termini sono stati assunti negativi, salvo quelli di peso zero corrispondenti alle variabili fittizie introdotte nelle relazioni vincolari, per compensare la conversione in uguaglianza.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16÷19	
1	1	1	1	1									1			...	≤100
2					1	1	1	1						1		...	≤ 90
3									1	1	1	1			1	...	≤ 80
4	1				1				1							...	=100
5		1				1				1						...	= 90
6			1				1				1					...	= 30
7				1				1				1				...	= 50
	x_{11}	x_{12}	x_{13}	x_{14}	x_{21}	x_{22}	x_{23}	x_{24}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}		

Figura A.7

Si hanno, quindi, 7 vincoli e 19 variabili di cui 3 fittizie e 4 di comodo. Dal prospetto dei risultati (figura A.8) si ricavano i valori delle incognite che minimizzano il costo complessivo di trasporto:

$$x_1 = 100; \quad x_6 = 90; \quad x_{11} = 30; \quad x_{12} = 50$$

La tabella operativa di marcia risulta:

		1 FI	2 RM	3 NA	4 PA	
1	MI	100 (1)				100
2	GE		90 (6)			90
3	BA			30 (11)	50 (12)	80
		100	90	30	50	270

```
>-----> RISULTATI DEL PASSO ------> N. 5 :
C. F. OBB. -4  -8  -10  -16  -5  -7  -8  -15  -12  -10  -8  -11   0   0   0
-10000  -10000  -10000  -10000
variabili  1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16
 17  18  19
 13   0   0   1   1   1  -1   0   0   0  -1   0   0   0   1   0   0  -1   0
0  0
 14   0   0  -1   0   0   1   0   1   1   0  -1   0   0   0   1   0   0  -1
0  0
 15   0   0   0  -1  -1   0   0  -1  -1   1   1   0   0   0   0   1   0   0
-1  -1
  1  100   1   0   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0
0  0
  6   90   0   1   0   0   0   1   0   0   0   1   0   0   0   0   0   0   1
0  0
 11   30   0   0   1   0   0   0   1   0   0   0   1   0   0   0   0   0   0
1  0
 12   50   0   0   0   1   0   0   0   1   0   0   0   1   0   0   0   0   0
0  1

F.OB. -1820   0  -1  -2  -5  -1   0   0  -4  -8  -3   0   0   0   0   0  -9996
-9993  -9992  -9989
I CALCOLI SONO TERMINATI. Tutto daccapo? (s/n)
```

Figura A.8

Al problema dei trasporti, come esempio per introdurre le tecniche della programmazione lineare, potremmo aggiungerne altri; ne elenchiamo alcuni, solo attraverso i loro enunciati, rinviando alle pubblicazioni di Ricerca Operative per una trattazione piu' sistematica.

IL PROBLEMA DI ASSEGNAZIONE DI RISORSE. A questa classe di problemi appartiene quello di un'azienda agricola in cui si debba scegliere come ripartire il terreno disponibile a un certo numero di colture, in modo da massimizzare i benefici della produzione, tenendo conto dei numerosi coefficienti e vincoli tecnici e economici; tra questi le ore di lavoro necessarie per ogni coltura, il totale delle ore di lavoro disponibili in un opportuno periodo di tempo (la stagione, l'anno...), il numero delle macchine agricole disponibili, il capitale massimo disponibile nel periodo per acquistare sementi, concimi e assumere mano d'opera.
Un altro problema di assegnazione di risorse e' quello della distribuzione ottimale di risorse idriche a un certo numero di utenze in modo da sfruttare le disponibilita' delle varie sorgenti del bacino al minimo costo, con il vincolo di soddisfare il fabbisogno minimo di ciascun destinatario.
IL PROBLEMA DELLE DIETE. Assegnati un certo numero di alimenti di cui sono noti i contenuti nutritivi, determinare al minimo costo, le quantita' di ciascun alimento in modo che non vengano superati certi limiti delle sostanze componenti (proteine, grassi, sodio, etc) tenendo anche conto del fabbisogno energetico minimo. Il problema puo' essere naturalmente molto meno riduttivo e considerare le necessita' individuali del paziente su un opportuno periodo, con schemi bilanciati di trattamento dietologico che tengano conto delle condizioni fisiologiche e degli eventuali squilibri nutritivi preesistenti.

A.3 ELEMENTI DI STATISTICA

E' nello spirito del libro creare molte occasioni d'incontro tra concetti teorici e loro traduzione in strumenti matematici di immediata utilita'. Nei pochi elementi di statistica qui tratteggiati si e' cercato di avvicinare gli argomenti scrivendo un opportuno programma **-strand3-** che calcola alcuni parametri statistici fondamentali. La spiegazione dei vari passi di tale programma e soprattutto i risultati ottenibili dalla sua esecuzione consentono di verificarne la logica e, implicitamente, l'avvenuta acquisizione dei concetti in esso contenuti.

A.3.1 ALCUNI RICHIAMI DI CONCETTI STATISTICI

Un insieme completo di osservazioni sul quale e' basata l'analisi statistica e' di solito chiamato un campione N, dove N e' il numero delle osservazioni. Il campione viene assunto per rappresentare un numero piu' grande di osservazioni o misure fatte nelle stesse condizioni ambientali.

Un gruppo molto piu' grande di possibili osservazioni viene chiamato **popolazione.** Tanto le misure appartenenti al gruppo della popolazione, quanto quelle del campione sono distribuite, in un certo modo, da un valore minimo a un valore massimo. Si definisce **campo di variazione** l'intervallo in cui cadono i valori sperimentali misurati. Il modo in cui tali valori si distribuiscono nel campo viene studiato attraverso opportuni parametri statistici che ne interpretano l'eventuale tendenza alla **centralita'** e ne definiscono il grado di **dispersione.**

La tendenza alla centralita' viene calcolata attraverso la media, espressa come

$$\bar{a} = (1/N)\sum_{i=1}^{N} a(i) \quad ,$$

dove i=1, 2, ... sono i numeri d'ordine delle singole misure.

Altre misure di centralita' sono la mediana o valore centrale della serie **ordinata** delle misure e la **moda**, o quella misura che ha la maggiore frequenza nella distribuzione.

Per la misura della dispersione si usano, in generale due misure: 1) il campo di variazione a(N)-a(1) dove a(1) e a(N) sono gli elementi corrispondenti ai valori minimo e massimo delle misure nel campo; 2) la deviazione standard, espressa dalla formula

$$DS = \frac{\left[\sum_{i=1}^{N} [a(i)-\bar{a})]^{2} \right]^{1/2}}{N}$$

A.3.2 MISURE DI CENTRALITA'

Interpretare una serie di dati numerici significa associar loro un un'idea in base a un insieme di regole che servano di standard per tutti, in modo da conservare la massima applicabilita'. Le misure di centralita' (media, mediana e moda) sono esempi, tra i piu' comuni, di strumenti per l'interpretazione di una serie di dati raccolti su un certo fenomeno.

Nelle prime linee (10÷30) del programma **strand3** si definiscono il numero degli elementi e il numero degli intervalli in cui catalogare i numeri introdotti con la linea 50.

```
10 'STRAND3
15 CLS:KEY OFF:INPUT"RISOLUZIONE GRAFICA=",Z:SCREEN Z
20 WINDOW (0,0)-(639,399):PRINT"ELEMENTI DI STATISTICA     ";
25 INPUT;"  N. ELEMENTI-->",N:INPUT ";  N. INTERVALLI-->",C
30 DIM V(N),F(N),T(N),AM(N),TM(N),S(N),Q(N),W(C+2),G(C+2),A(N+C),Z(N+C)
40 PRINT:FOR E=1 TO N
45 PRINT "  V("E")=";
50 INPUT;" ",V(E):PRINT";";
```

Durante l'introduzione degli N elementi, questi vengono caricati, contestualmente, in un vettore V(E) insieme alla creazione di un vettore F(E), che serve per l'ordinamento indiretto del vettore V [F(E)=E]. Per la tecnica dell'ordinamento si veda il programma **ordind** al capitolo 9. All'ultimo ciclo della linea 220 il vettore degli indici F(E) e' ordinato secondo i contenuti del vettore V. Si noti che le inversioni sono fatte sugli indici.

```
60 F(E)=E:NEXT
100 FOR P=1 TO N-1
110 T=0
130 FOR E=1 TO N-P
140 IF V(F(E))<=V(F(E+1)) THEN 200
160 Q=F(E)
170 F(E)=F(E+1)
180 F(E+1)=Q
190 T=1
200 NEXT E
210 IF T=0 THEN 240
220 NEXT P
240 PRINT
```

Il vettore ordinato e' stampato al ciclo 250÷270, mentre alle linee 500÷530 si crea un vettore A(E), i cui elementi sono disposti in ordine crescente come gli indici. Dell'ordinamento del vettore A si approfitta subito per calcolare la **mediana**. Il calcolo prevede due strade: 1)quella in cui essendo dispari il numero degli elementi individua la mediana come elemento centrale della serie (meta' numeri a sinistra, meta' a destra); 2)quella in cui, essendo pari il numero degli elementi, si assume come valore mediano la media dei due valori piu' centrali della serie. Ad esempio, nella serie crescente (1 2 3 4 12 20) la mediana e' (3+4)/2=3.5.

Il confronto alla linea 615 accerta se N e' pari o dispari. Se e' dispari la parte intera di N/2 non sara' uguale a N/2 e il numero d'ordine dell'elemento centrale sara' quindi uguale a tale parte intera piu' uno.

Il calcolo della **moda** sfrutta il fatto che la serie e' gia' ordinata in ordine crescente. Il vettore T(x) registra nell'ordine tutte le volte in cui un elemento compare piu' di una volta **(la linea 675 rileva appunto se un elemento A(E) e' seguito o no da un elemento uguale**: in tal caso il contenuto di T(x) si incrementa di un'unita'). Il valore di **tale elemento**, che compare piu' di una volta nella serie, **e' caricato nel vettore AM(x)** alla linea 676. La linea 850, nel corso di una FOR...NEXT da 1 a N, stampa tutti i valori non nulli degli elementi che si ripetono piu' di una volta.

```
245 PRINT "ELEMENTI ORDINATI: ";
250 FOR E=1 TO N
260 PRINT V(F(E));
270 NEXT E
500 FOR E=1 TO N
510 A(E)=V(F(E))
530 NEXT
600 'mediana -------------------------
610 N1=INT(N/2)
615 IF N1<>N/2 THEN NC=A(N1+1):GOTO 650
620 NA=A(N1)
630 NB=A(N1+1)
640 NC=(NA+NB)/2
650 'MODA    -------------------------
653 X=1
655 FOR E=1 TO N
670 FOR H=E TO N
675 IF A(E)<>A(H+1) THEN E=H:GOTO 684
676 T(X)=T(X)+1:AM(X)=A(E)
680 NEXT H
684 X=X+1
685 NEXT E
686 FOR X=1 TO N
688 NEXT
```

Alla linea 710 si accumulano nella variabile S i valori della serie, cioe'

$$S = \sum_{e=1}^{N} a(e) \qquad (1)$$

La media M = S/N e' calcolata alla linea 750.

```
700 FOR E=1 TO N
705 U=U+(A(E))^2
710 S=S+A(E)
720 NEXT
750 M=S/N
```

A.3.3 MISURE DI DISPERSIONE

Alla linea 750 si accumulano nella variabile U i quadrati dei valori della serie, cioe'

$$U = \sum_{e=1}^{N} [a(e)]^2 \qquad (2)$$

La varianza, o scarto quadratico dalla media, e' calcolata alla linea 760 con la formula

$$SQ = [U - (S\wedge 2)/N]/N \qquad (3)$$

La formula che calcola SQ equivale alla formula

$$SQ = (1/N)[\sum_{e=1}^{N}(a(e)-M)\wedge 2] \qquad (4)$$

In realta' la formula si riferisce al caso in cui N e' molto elevato e quindi la media $\bar{a}$=M si avvicina molto alla media dei valori della popolazione. Si puo' dimostrare che la somma delle deviazioni quadratiche dalla media aritmetica e' minore di quelle ottenute dalle deviazioni rispetto a un qualunque altro valore; cioe'

$$\Sigma[a(e)-M]^2 < \Sigma[a(e)-\mu]^2$$

La disuguaglianza esprime il fatto che quando si stima la varianza con M, anziche' con μ, si ha un valore minore della varianza. L'errore viene sufficientemente corretto se si sostituisce N-1 a N, cioe' se si stima la varianza con la formula

$$SD = [U-(s\wedge 2)/N]/(N-1)$$

La modifica al programma (linea 765) avviene moltiplicando SQ per N/(N-1). Il divisore (N-1) rappresenta il numero di gradi di liberta' nella stima della varianza, cioe' il numero di confronti indipendenti che possono essere fatti con N osservazioni. Infatti, i confronti eseguiti nella (4) sono [a(1)-M], [a(2)-M],...,. Soltanto N-1 di essi sono indipendenti, poiche' M e' calcolato dalle osservazioni e se sono noti M e (n-1) valori di a e' possibile determinare l'ultimo. La varianza misura la dispersione dei dati se e' nota la vera media della popolazione μ: in tal caso, il numero dei gradi di liberta' e' uguale al numero delle osservazioni.

```
760 SQ=(U-(S^2)/N)/N
765 SD=SQ*N/(N-1)
770 PRINT:PRINT
790 PRINT "CAMPO DI VAR.="A(N)-A(1)
800 PRINT "MEDIA DEGLI N num.:"M
810 PRINT "MEDIANA="NC
820 FOR X=1 TO N
850 IF T(X)<>0 THEN PRINT "MODA("X")="AM(X) "("T(X)+1"volte)"
860 NEXT
```

```
862 PRINT:FOR X=1 TO N':PRINT "T("X")="T(X)" ";
863 FOR XX=1 TO N
865 IF T(X)>T(XX) THEN 868
867 NEXT
868 MM=AM(X):ZT=T(X)+1
870 NEXT X'PRINT "MAXMODA  ="MM" ("ZT"volte)"
875 FOR X=1 TO N':PRINT T(X);
876 IF T(X)<>0 THEN Y=Y+1:TM(Y)=T(X)+1
878 NEXT
880 FOR Y=1 TO N-1
881 FOR YY=Y TO N
882 IF TM(YY)=0 THEN 884
883 IF TM(Y)=TM(YY+1) THEN 900
884 NEXT
885 NEXT
886 IF ZT=1 THEN 900
888 PRINT "MAXMODA="MM"("ZT"volte)"
900 PRINT:PRINT "VARIANZA (s^2) :"SQ
950 PRINT "VARIANZA(n/n-1):"SD
970 PRINT "DEVIAZIONE STD.:"SQR(SQ)
1000 D=(A(N)-A(1))/C
1100 FOR I=2 TO C:G(1)=A(1)
1110 G(I)=G(I-1)+D:NEXT
1125 PRINT "ESTR.SIN./inter. = ";
1130 FOR I=1 TO C
1140 PRINT G(I);:NEXT:PRINT
1160 QQ$=INPUT$(1)
```

Dalla linea 1000 ha inizio il trattamento dei dati per la selezione dei valori che rientrano in uno determinato intervallo; alla linea 1000, in particolare, si definisce l'ampiezza (D) dell'intervallo in funzione del campo totale di variazione dei valori osservati (che dopo l'introduzione iniziale si trovano registrati nel vettore A) e il numero (C), scelto all'inizio, di intervalli in cui raggruppare i valori stessi. L'inserimento dei valori a(e) nei singoli intervalli avviene mediante un vettore W dedicato e il criterio viene regolato dai confronti scritti alle linee 1232 e 1235. I valori G(i) degli estremi sinistri dei diversi intervalli vengono stampati alla linea 1140.

```
1200 FOR E=1 TO N
1210 FOR I=1 TO C
1225 IF A(E)<=G(I) THEN 1235
1230 NEXT I
1232 IF A(E)>G(C) THEN W(C)=W(C)+1:GOTO 1240
1235 IF A(E)=G(I) THEN W(I)=W(I)+1 ELSE W(I-1)=W(I-1)+1
1240 NEXT E
```

Piu' precisamente, la selezione degli elementi avviene mediante due cicli, uno all'interno dell'altro: il primo, sotto il controllo della variabile di ciclo E ha inizio alla linea 1200 e consente di prendere un elemento alla volta della serie a(e) e affidarlo al controllo del ciclo interno gestito dalla variabile [I] che evolve fino a [C]. Alla linea 1225 si "pesa" il valore confrontandolo con l'estremo sinistro

dell'intervallo corrente e solo se il valore della pesata risulta superiore a tale estremo si passa al confronto coll'estremo immediatamente a destra. Quando l'elemento in esame superasse anche l'estremo destro del penultimo intervallo [a(e) > G(c)] allora significherebbe che deve appartenere all'ultimo intervallo. L'evento viene registrato sommando "1" al vettore (c) che realizza fisicamente il contenitore degli eventi selezionati. Esaurita l'analisi di un elemento, si passa a quella dell'elemento successivo (NEXT E). Quando il confronto alla linea 1225 indica che il valore **non** supera l'estremo destro dell'intervallo occorre ancora vedere se coincide o se e' inferiore. Nel primo caso si somma "1" alla variabile W che porta l'indice corrispondente, altrimenti (else) a quella immediatamente prima. La figura A.9 illustra la situazione delle variabili in gioco attraverso un esempio di serie. Si noti che la somma dei valori di W e' pari a quella degli **elementi** N (non valori). Allora Z(E)=W(E)/N rappresenta la frequenza dell'evento.

```
1245 PRINT "N. EVENTI PER INTERV. =";
1250 FOR I=1 TO C
1260 PRINT W(I);
1270 NEXT
1280 QQ$=INPUT$(1):CLS
5000 'liret
5005 CLS
5135 PRINT "FREQ.Z(e) ";
5140 FOR E=1 TO C
5150 Z(E)=W(E)/N:PRINT Z(E);:NEXT
```

Figura A.9

Un'esecuzione di **strand3** con N (numero degli elementi)=8, C (numero degli intervalli)=5 e con la stessa serie di numeri conduce agli stessi risultati, come qui di seguito riportato.

```
RUN
RISOLUZIONE GRAFICA=3
ELEMENTI DI STATISTICA        N. ELEMENTI-->8;  N. INTERVALLI-->5

  V( 1 )= 1;  V( 2 )= 2.46;  V( 3 )= 4;  V( 4 )= 4.5;  V( 5 )= 5;  V( 6 )= 5.2;
  V( 7 )= 8;  V( 8 )= 8.3;
ELEMENTI ORDINATI:  1  2.46  4  4.5  5  5.2  8  8.3

CAMPO DI VAR.= 7.3
MEDIA DEGLI N num.: 4.8075
MEDIANA= 4.75


VARIANZA (s^2) : 5.416895
VARIANZA(n/n-1): 6.190737
DEVIAZIONE STD.: 2.327423
ESTR.SIN./inter. =  1  2.46  3.92  5.38  6.84
N. EVENTI PER INTERV. = 1  1  4  0  2
FREQ.Z(e)  .125  .125  .5  0  .25
```

La rappresentazione in istogramma del vettore Z(e), che si avvale del programma **liret** opportunamente ritagliato allo scopo, conclude il programma.

```
5170 QQ$=INPUT$(1):CLS
5200 INPUT;"  PASSO ORIZZONTALE --->",P
5210 INPUT ";  BASE ISTOGRAMMA --->",B
5213 CLS:LINE (8,0)-(8,399),1
5215 FOR J=0 TO 380 STEP 10:LINE (6,J)-(8,J):NEXT
5216 FOR J=0 TO 380 STEP 100:LINE (3,J)-(8,J):NEXT
5217 LOCATE 1,1:PRINT "1"
5230 FOR K=1 TO C
5231 LINE (10+(K-1)*P/N,0)-(10+P/N+(K-1)*P/N,0),1,B
5232 LINE (10+(K-1)*P/N,0)-(10+B/N+(K-1)*P/N,1),1,B
5235 LINE (10+(K-1)*P/N,0)-(10+B/N+(K-1)*P/N,399*(Z(K))),1,B
5238 NEXT
5340 QQ$=INPUT$(1):GOTO 5005
```

Si noti che per una migliore leggibilita' e' possibile ridefinire il passo orizzontale e la base dell'istogramma. Nella figura A.10 la stessa serie di dati e' rappresentata mediante frequenze.
Nella figura A.11 abbiamo un altro esempio di analisi statistica su una serie di elementi, molti dei quali ripetuti.
Per motivi di leggibilita', il programma contiene le protezioni minime sull'introduzione dei dati: il lettore esigente puo' introdurne altre per le proprie esigenze personali. Altre migliorie suggerite riguardano la logica di introduzione dei dati: anziche' definire il numero degli elementi di cui e' costituito il campione e successivamente decidere l'ampiezza dell'intervallo in cui raggruppare i singoli dati, puo' essere conveniente definire prima il numero degli intervalli, in modo che il raggruppamento avvenga all'atto dell'introduzione.

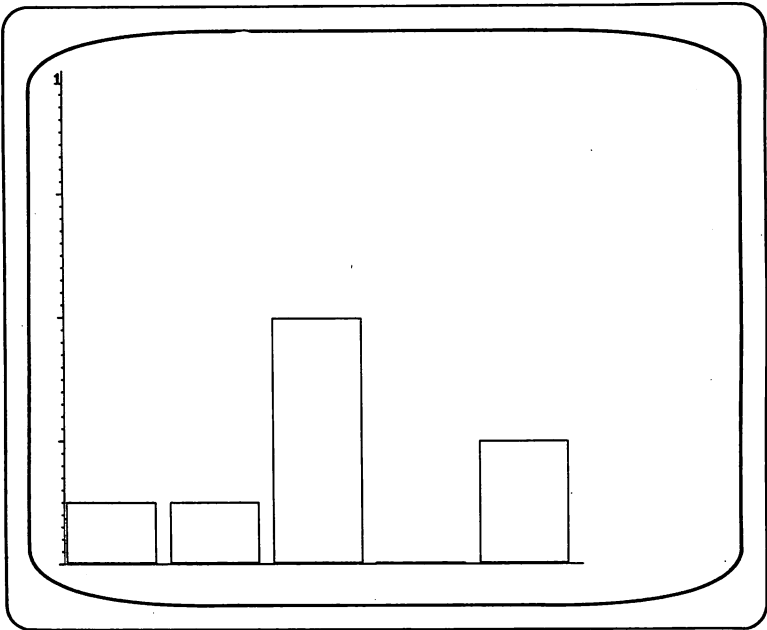



Figura A.10

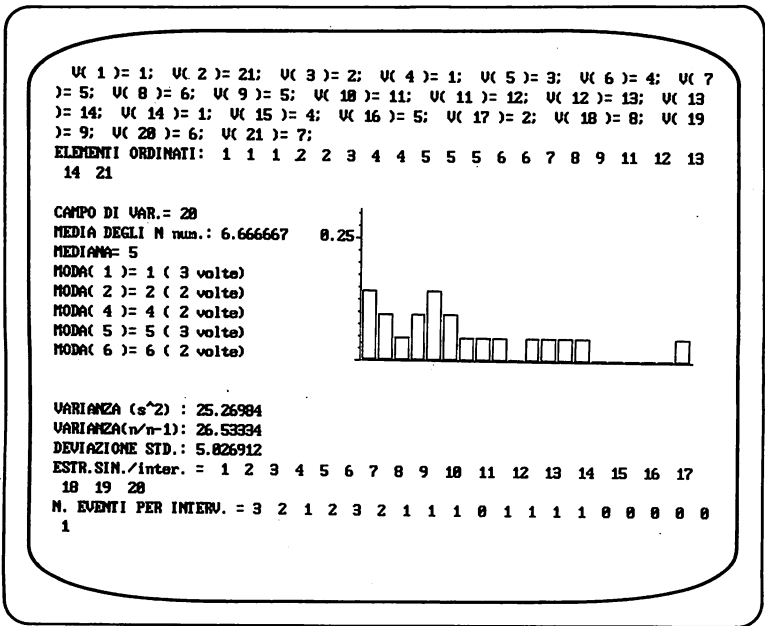



Figura A.11

UN MINIMO DI MS-DOS

Che cosa sia un sistema operativo e' gia' stato detto nella prima parte del libro: la' abbiamo cominciato a usare questo intermediario tra le risorse della macchina e il linguaggio Basic, uno dei tanti componenti disponibili del sistema. Ma l'uso di MS-DOS non si limita solo alla disponibilita' di semplici comandi per il richiamo dei singoli componenti. Esiste anche un modo piu' sofisticato di impiegarlo, attraverso il quale si puo' definire un ambiente operativo particolare capace di assegnare alla macchina ruoli nuovi di automazione per sollevare l'utente da complicate e ripetitive sessioni di attrezzaggio.

Anche solo per arrivare alla definizione di un primo livello di automazione e', tuttavia, necessario riprendere i concetti piu' elementari, dai primi segni di presenza di MS-DOS, quando si accende la macchina, fino alla conoscenza dei comandi essenziali e alle regole sintattiche che li governano.

B.1 LA MACCHINA E IL DISCO SISTEMA

Ci riferiamo alla macchina base illustrata nelle prime pagine del libro: unita' centrale M24, due unita' a floppy disk denominate **a** (inferiore) e **b** (superiore), un video monocromatico e una stampante PR 15B. Su macchine compatibili il drive **a** e' quello di sinistra. Nell'unita' **a** va inserito il disco sistema contenente MS-DOS. Solo cosi' puo' avvenire il caricamento in memoria centrale del sistema operativo. E' inutile pensare di attuare tale caricamento inserendo il dischetto in **b** (unita' superiore) perche' **all'accensione la macchina da' ascolto solo all'unita'** [a]. La conferma di cio' si ha dal segnale di riconoscimento di MS-DOS [>] che segue la lettera **a**.
A> significa quindi in modo molto conciso:

"MS-DOS presente in attesa di cercare comandi **esterni** sul dischetto inserito nel drive **a**".

E' opportuno sottolineare che qui il significato di drive attivo sta per predisposto ad operare, preselezionato (sempre che abbia un dischetto su cui lavorare e un comando che lo coinvolga).

Per scambiare le funzioni del drive **a** con quelle del drive **b** basta introdurre: [b:] **CR** e la disponibilita' di MS-DOS da quel momento avverra' mediante il drive **b.** Infatti ora il segnale [>] di presenza segue il nome del drive **b.** E la disponibilita' dei componenti MS-DOS contenuti nel dischetto sistema si potra' avere **solo** se esso sara' stato inserito nel drive **b.** Per tornare allo stato [a>] basta nuovamente introdurre [a:] **CR.** Quando un drive assume la prerogativa di essere implicitamente selezionato viene detto in gergo informatico **drive di default.**

Abbiamo appena visto come tale proprieta' possa essere scambiata attraverso l'invio di un semplice messaggio da tastiera, anziche' l'uso di un interruttore, come qualcuno di noi avrebbe forse preferito. Vediamo ora le tre parti essenziali e visibili di cui MS-DOS e' costituito. Queste sono:

1. Il caricatore (BOOTER)
2. Il nucleo (COMANDI INTERNI)
3. L'elaboratore di comandi (COMMAND PROCESSOR)

Tutti e tre questi componenti risiedono nel dischetto sistema, senza il quale il calcolatore non puo' funzionare. Il caricatore viene letto dal disco MS-DOS e portato in memoria centrale. Da qui si attiva l'operazione che porta in memoria dei comandi che vi rimarranno fino allo spegnimento della macchina. (Tali comandi vengono percio' detti interni o residenti).

La terza parte del DOS e' l'elaboratore di comandi che interpreta ed elabora le richieste di comandi fatte dall'utente. Alcune di queste richieste possono riferirsi a semplici scambi di assegnazione del drive di default [A>b:] [B>a:], oppure ai comandi interni gia' presenti in memoria, oppure, infine, ai comandi non residenti, detti anche transienti, che vengono trasferiti dal disco DOS in memoria, nella quale sostano per il tempo necessario a svolgere le funzioni assegnate.

MANOVRE SPECIALI IMPIEGABILI DURANTE LE SESSIONI DOS

Sono disponibili le manovre gia' note come:

CTRL-NUMLOCK	interrompe le operazioni fino alla digitazione di un qualsiasi tasto. (Serve ad esempio per interrompere lo scorrimento verticale su video di un lungo testo che altrimenti andrebbe rapidamente perduto oltre la sommita' dello schermo);
SHIFT-PRTSCR	per hard copy, cioe' per ottenere la copia su stampante del contenuto del video;
CTRL-PRTSCR	per avere un'eco stampata di tutto quanto appare su video.

Una volta caricato stabilmente in memoria il nucleo di MS-DOS dal dischetto la macchina puo' fare a meno del dischetto sistema, solo per i comandi interni al nucleo che sono: COPY, DATE, DIR, ERASE, PAUSE, REM, TIME, e TYPE. Per tutti gli altri, essendo esterni al nucleo, necessita la presenza del dischetto sistema che li contiene, da cui estrarli, di volta in volta, quando l'utente ne ha bisogno.

B.1 I PRINCIPALI COMANDI DOS. REGOLE E CONVENZIONI.

FORMAT e' uno dei primi comandi da imparare, perche' consente di predisporre un dischetto vergine in modo che possa accogliere i dati da registrare secondo uno standard prefissato.

L'operazione coinvolge tutto il dischetto e ne distrugge eventuali contenuti. Per questo si dice che il comando inizializza o "formatta" il dischetto.
Poiche' FORMAT non e' un comando interno occorre che al momento in cui viene dato (battendone il nome sulla tastiera con la conferma del tasto [return]) il dischetto sistema sia regolarmente inserito nel drive **a** (quello inferiore). L'operazione normalmente avviene ponendo il disco vergine (o da riformattare) nel drive **b** e inizia quando, dopo aver battuto **format b:**, si preme il tasto **return.** La durata dell'operazione e' di circa 50 secondi e al termine il disco sara' strutturato con 9 settori per ciascuna delle 32 tracce delle due facce. Esistono diverse varianti della FORMAT: elenchiamo le piu' importanti:

```
format b:     formatta a 9 settori il disco inserito nel drive b
format b:/8   formatta a 8 settori il disco inserito in b
format b:/s   formatta a 9 il disco in b e vi incide anche il nucleo
```

Si noti che quando e' attivo il drive **b**, B>**a:format** formattera' il disco in b con il comando estratto dal drive in a!

Per creare questa situazione di lavoro si accenda la macchina avendo posto un disco vergine in **b** e il DOS in **a.** Appena appare il messaggio a> digitando **b:** [CR] ordiniamo a MS-DOS che nomini operativo il drive in **b**; infatti sul video apparira' b> al posto di a>.

A questo punto se digitiamo **a:format [CR]** la risposta sara':

```
Insert new diskette for drive b:
and strike any key when ready
```

che significa: l'ordine **a:format** si riferisce a un comando contenuto nel corredo DOS che si trova nel dischetto in **a,** ma operera' sul dischetto inserito nel drive attivo (cioe' il **b:**).

Dopo queste necessarie informazioni sulla struttura fisica del supporto magnetico e la sua preparazione vediamo le regole che occorre rispettare per effettuare una lettura o una registrazione di dati su dischetto.

Innanzitutto definiamo il contenuto di un dischetto in termini di informazioni organizzate. Si chiamano **file** le unita' logiche di informazione in cui un dischetto puo' essere suddiviso. Esempi di file sono le sezioni in cui questo libro e' stato suddiviso nell'operazione di text editing.

Questa appendice e' una di esse e il relativo nome del file che lo registra su dischetto e' **h1.** Tra le regole da rispettare con i file esistono quelle per dar loro un nome. Il nome di un file e' costituito di due parti.

- il nome vero e proprio (max 8 caratteri)
- estensione o sottonome (eventuale) (max 3 caratteri)

Le due parti devono essere separate da un punto.

Esempio: h1.txt

Per formare un nome si possono usare:

- le 26 lettere dell'alfabeto inglese dalla A alla Z
- le cifre da 0 a 9
- i simboli come & $ % # @ ! - ~ ( ) } { '

Non e' consentito impiegare i seguenti simboli:

+ * / = > < : , ? \ ÷ ne' il carattere bianco o spazio.

Comunque si scrivano le lettere dell'alfabeto di cui e' costituito il **nome**, saranno sempre **registrate in maiuscolo.**

Sullo stesso dischetto, ovviamente, non possono esserci due file diversi con lo stesso nome per il semplice fatto che il nome e' l'unica chiave di accesso di un file. E' altrettanto ovvio che due file diversi possano avere **lo stesso nome su due dischetti diversi.**

OGNI DISCHETTO HA LA SUA DIRECTORY (INDICE DEI NOMI)

Per vedere rapidamente cosa c'e' in un dischetto si puo' consultarne l'indice (detto directory) col comando interno DIR. Basta scrivere a segnale di DOS presente:

A> **dir a:** per il dischetto in **a**
A> **dir b:** per il dischetto in **b**

Per il drive corrente (di default) basta fare **dir.**
Se la lista non sta interamente sul video o si stampa in tandem avendo fatto la manovra **CTRL PrtSCR** (tenendo premuto il tasto speciale CTRL battere il tasto PrtSC) prima di battere **dir**. Oppure, si blocca lo scorrimento con la manovra **CTRL NUMLOCK.** Oppure si da' il comando per la visualizzazione ridotta:

dir a:**/w** oppure dir b:**/w** oppure dir **/w**

Una delle prime cose che l'utente cerca e' come copiare un file: occorre ovviamente conoscerne il nome e il dischetto in cui si trova. E poi occorre saper manovrare il comando giusto nel modo corretto. Il comando in causa e' **copy** e ne potrete vedere il nome e sottonome (.com) facendo una **dir** sul dischetto sistema.
Per fare la copia su un dischetto (che deve essere gia' formattato) posto nel drive b [b:] di un file registrato col nome **pluto** sul dischetto posto in **a:**, mantenendogli lo stesso nome, bisogna lanciare il seguente comando:

copy a:pluto b:

Se, nel trasferimento, vogliamo anche ridenominarlo pluto1, occorrera' specificarlo nell'etichetta dopo l'indicazione del drive di destinazione. Cosi':

copy a:pluto b:pluto1

Alcuni non si fidano troppo degli strumenti informatici e preferiscono usare un comando che li tranquillizzi. Si tratta del comando **comp**, con il quale il contenuto di un file viene confrontato col contenuto di un altro file.
Cosi' volendo verificare se i due file pluto e pluto1, il secondo ottenuto per copia dal primo, sono effettivamente identici, basta scrivere:

comp a:pluto b:pluto1

Avvertenza importante: separare con degli spazi il nome del comando da cio' che segue e la fine del nome del file da cio' che segue; inoltre **non** introdurre spazi in testa al nome del file.
I seguenti comandi

```
compa:pluto  b:pluto1
com a:  pluto  b:pluto1
comp a:plutob:pluto1
```

non verrebbero accettati perche':

1) non esiste un comando **compa:**
2) lo **spazio** e' un carattere illegale per i nomi dei file
3) probabilmente non esiste il nome **plutob.**

B.3 NOMI DI PERIFERICHE

La necessaria precisione a specificare il drive nel quale e' inserito il dischetto in cui si vuole ricercare un file -come farebbe la macchina a trovarlo se inserissimo il dischetto nel drive sbagliato?- ha spinto a generalizzare il concetto di sorgente e destinatario di un file, e a meglio definire il processo che, permettendo un flusso tra sorgente e destinatario, consenta il trasferimento di un file. Cosi' sono nati i nomi delle periferiche.
Esistono i nomi delle periferiche drive **a: b: c: d:** (**c** e **d** sono i nomi degli hard disk integrato e esterno)? Per la stessa ragione esistono i nomi di altre periferiche, come:

con:	console, sta per tastiera
scrn:	screen, sta per schermo o video
lpt1:	stampante numero 1
lpt2:	stampante numero 2
com1:	adattatore per comunicazioni n. 1
com2:	adattatore per comunicazioni n. 2

L'insieme del nome del file e quello della periferica interessata deve esprimere senza ambiguita' dove sia il file su cui desideriamo operare. Ad esempio **a:h1** si riferisce a un file di nome **h1**; la manovra interessera' il drive **a** (e il dischetto che vi e' inserito).

B.4 NOMI ABBREVIATI

La macchina puo' fare una ricerca anche solo basandosi sui sottonomi. Ad esempio, con la notazione ***.ico** si intendono tutti i file con sottonome **ico**, mentre con **oli.*** tutti i file di nome **oli**, qualunque estensione abbiano. Con l'espressione ***.*** tutti i file.
L'impiego dei nomi abbreviati puo' far risparmiare tempo, ma vanno usati con cautela, perche' la fretta puo' indurre in errori irreparabili. Ad esempio con l'operazione **copy a:*.* b:** si interessano tutti i file contenuti nel dischetto **a** e si trasferiscono col loro nome sul dischetto posto in **b:**. Operazione altrettanto rapida, ma delicata, quella che anziche' copiare files li distrugge! Si tratta, ad esempio, del comando **erase b:ico.*** che cancella su dischetto posto in b: tutti i file di nome **ico**, qualunque sottonome abbiano.
Un altro esempio interessante, che ci offre anche la possibilita' di imparare il comando **copy**, e' il seguente:

copy **a:***.ico **b:**	trasferisce in **b** tutti i file con estensione **ico**, presenti sul dischetto posto in **a**.

Un altro artificio stenografico e' l'impiego del carattere [?]. Lo si puo' inserire al posto di qualsiasi carattere di un nome per indicare l'indifferenza al carattere che sostituisce. Ad esempio:

copy a:**oli??.bas** b:	trasferisce da **a** a **b** tutti i file i cui nomi cominciano con **oli** e hanno estensione **bas**.

B.5 COME CREARSI UN FILE SU DISCHETTO

Con il comando interno **copy** e' possibile trasferire direttamente i dati su un dischetto mentre si digitano da tastiera. Descriviamo le operazioni che possono essere suddivise nei seguenti passi:

- Si dia un nome al testo che si vuol trasferire e che diventera' il nome del file (promem).

- Si definisca il drive interessato al trasferimento (a:).

- Si scriva il comando cosi': **copy con: a:promem** [CR]

- Si batta il testo, facendo attenzione che per andare a capo alla riga seguente bisogna usare il tasto CR.

- Finito il testo, che fin qui e' ancora in memoria, per registrarlo basta premere il tasto funzionale **F6** seguita da CR.

L'ultimo passo va eseguito dopo aver portato il cursore a inizio riga; o meglio, la manovra di chiusura del testo (F6+CR) e inizio delle operazioni di scrittura su dischetto ha effetto solo se battuti all'inizio della riga (dopo un return).
Per verificare se effettivamente la registrazione e' avvenuta basta

rileggere il file promem con l'operazione inversa:

copy a:promem con:

La rilettura puo' essere anche fatta col comando:

type promem o piu' precisamente **type a:promem**

La stampa del file promem viene eseguita lanciando il comando:

copy promem lpt1: o piu' precisamente **copy a:promem lpt1:**

Altri comandi immediati:

A> erase a:promem — per cancellare il file promem sul dischetto posto in **a**

A> rename b:promem b:antmem — (promem: vecchio nome)

A> data 5-12-1990 — rimette il datario di macchina

A> time 12:12:12 — rimette l'orologio di macchina

A> chkdsk b: — consente di esaminare l'occupazione del dischetto posto in **b:**, supponendo il DOS montato nel drive attivo **a:**

A> b: chkdsk a: — consente di avere la situazione occupazione del dischetto montato in **a:**, avendo il DOS montato in **b:**

A> chkdsk — fornisce l'occupazione del dischetto sistema posto in **a:**; se fosse posto in **b:** non si otterrebbe niente.

A> diskcopy a: b: — copia l'intero contenuto del dischetto posto in **a:** sul dischetto posto in **b:** il quale perde completamente il precedente contenuto. Due avvertenze importanti: la prima e' che il dischetto che riceve la copia puo' non essere formattato in quanto provvede gia' il comando diskcopy a farlo. La seconda e' che, una volta lanciato il comando con il dischetto DOS posto in [a:], questo va sostituito con il dischetto che si vuol copiare, altrimenti faremo la copia del dischetto DOS.

A> diskcomp a: b: — confronta l'intero contenuto del dischetto contenuto in a: con l'intero contenuto di quello posto in b:

mode lpt1:132,6 — il comando e' diretto alla stampante collegata (nel nostro caso supponiamo la PR 15B) e fa si' che la stampa avvenga con righe di 132 caratteri. con una spaziatura orizzontale di 16,6 caratteri per pollice e una verticale di 6 righe per pollice.

graphics — questo comando va attivato prima di effettuare la manovra con i tasti SHIFT PrtSC per fare in modo che venga stampata su stampante grafica (tipo PR 15B) l'immagine video ottenuta in ambiente grafico con le varie risoluzioni (screen 1, 2, 3,).

CHKDSK, TIME, DATE, FORMAT, DISKCOPY MODE e GRAPHICS non sono comandi che fanno parte del nucleo di DOS e pertanto prima di usarli bisogna essere sicuri che il dischetto posto nel drive coinvolto dall'operazione li annoveri nella sua lista. Non intendiamo esaminare molti altri comandi DOS per mantenere semplice, per quanto e' possibile, la lettura di questo libro; esistono per l'opportuna completezza i manuali delle Case costruttrici. Nel prossimo paragrafo vedremo invece l'uso coordinato dei comandi gia' esaminati.

B.6 I SUPERCOMANDI

Chiamiamo **supercomandi** un insieme concatenato di comandi DOS che l'utente puo' costruirsi per i propri scopi specifici. Nella letteratura li trovate anche coi nomi di **file bat** o **batch file.**
Vediamo subito come costruirli, avendo in mente chiari lo scopo e l'utilita' per farlo.

Descriviamo, innanzitutto, un obiettivo e la procedura per realizzarlo senza l'uso dei supercomandi.

Supponiamo di voler fabbricare un supercomando per rendere un po' piu' automatica e guidata la sequenza delle operazioni per fare la copia di un dischetto: col termine guidata vogliamo riferirci alla possibilita' di introdurre, durante le operazioni, dei commenti esplicativi sui successivi passi e sugli effetti da essi prodotti insieme a delle pause per poter proporre delle scelte.

Commenti e pause si realizzano con due comandi interni, facenti parte del nucleo di DOS: si tratta dei comandi **rem** e **pause.** Una volta caricato il nucleo, questi due comandi restano residenti in memoria finche' non si spegne la macchina.

Per fabbricare un supercomando bisogna anche pensare a assegnargli un nome e inoltre a dargli l'estensione **.bat**: in tal modo il sistema sa che si tratta di un supercomando.

L'obiettivo sia di fabbricare un supercomando che esegua le seguenti operazioni:

1) copiare il contenuto del dischetto presente in **a** nel dischetto presente in **b**;
2) confrontare, dopo aver completato l'operazione in 1), i contenuti dei due dischetti.

Dati gli scopi cui il supercomando e' destinato sembra ragionevole chiamarlo **copia**.bat: vediamo come crearlo, insieme a quei commenti o avvertimenti di cui prima dicevamo.

La procedura di generazione impiega il comando **copy con:**, gia' descritto per la creazione di un file da trasferire direttamente da tastiera a dischetto. Inseriamo dunque un dischetto formattato con l'opzione [/s] -che contenga percio' il nucleo di MS-DOS- nel drive **a** (sia questo il drive attivo) e digitiamo in successione i seguenti comandi:

A> **copy con: copia.bat**
rem l'operazione conduce alla perdita dei file presenti sul dischetto in b
pause per procedere battere return, altrimenti ctrl-break
diskcopy a: b:
rem copia eseguita con confronto

Si osservi che il supercomando ha interessato il drive **a** e quindi viene registrato sul dischetto in **a**: evidentemente quando si vorra' eseguirlo dovremo fare in modo che il dischetto che lo contiene sia inserito nel drive **a** e che tale drive sia quello attivo (di default) al momento del lancio. Inoltre occorre che nel dischetto in cui abbiamo registrato **copia.bat** siano contenuti anche i comandi esterni DOS che il supercomando certamente vorra' avere disponibili, cioe' i comandi **diskcopy** e **diskcomp** (rem e pause fanno parte del nucleo e quindi sono sempre residenti in memoria fin dall'accensione della macchina). In caso contrario, se cioe' quei due comandi esterni non sono gia' nel dischetto, il supercomando non puo' inventarseli e si arrestera'.

Un'illustrazione del contenuto del dischetto potrebbe, quindi, risultare come quella riportata nella figura B.1.

COPIA .BAT	REM	PAUSE	DISKCOPY A: B:	DISKCOMP A: B:	REM

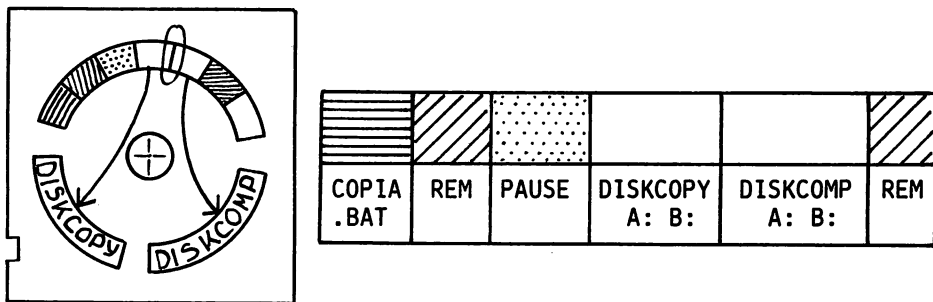



Figura B.1

A questo punto comincia a delinearsi come dovremo organizzare le cose e soprattutto come riformulare il problema di fare la copia di un dischetto descrivendo meglio le circostanze in cui tale copia deve essere fatta. Una situazione abbastanza ricorrente e' quella di chi sta creando dei programmi in Basic con un certo dischetto di lavoro che contiene il nucleo del DOS e alcuni dei suoi componenti piu' pregiati come gwbasic e graphics e voglia anche **salvare una certa situazione dei suoi programmi** su un altro dischetto. (Naturalmente si puo' fare una copia di un dischetto, usando di volta in volta i singoli comandi del DOS: con il supercomando, pero', la copia e' piu' automatica.)

Ci si puo' domandare se all'inizio occorra comunque mettere su quel dischetto di lavoro i vari componenti (gwbasic, graphics, diskcopy, diskcomp) e lo stesso supercomand copia.bat) e come bisogna procedere per ridurre il numero delle operazioni da fare, senza dimenticarne qualcuna, ogni volta che si attiva un nuovo dischetto destinato a un certo lavoro. Sara' sufficiente crearsi una versione personalizzata del dischetto DOS che contenga un supercomando un po' piu' raffinato di copia.bat, che rinomineremo **mater.bat.** L'obiettivo di mater.bat sia dunque quello di creare automaticamente il dischetto di lavoro posto nel drive **b:** contenente i componenti prima elencati. Per fabbricare e introdurre tale supercomando nel dischetto contenente il DOS useremo ancora la stessa procedura **copy con:** gia' utilizzata per la generazione di copia.bat. Ora, pero', l'organizzazione e' un po' piu' complessa e risulta cosi' articolata, insieme a oppurtuni commenti e pause (si ricordi di mettere il disco sistema in **a:**):

```
A>    copy con: mater.bat
      rem  attenzione! l'operazione formatta il disco in b e ne
      distrugge il contenuto
      pause  per procedere battere return, altrimenti ctrl-break
      format b:/s
      copy diskcopy.com b:
      copy copia.bat b:
      copy autoexec.bat b:
      copy graphics.com com b:
      copy gwbasic.exe b:
      copy chkdsk.com                               [F6+return]
```

Come si sara' notato con mater.bat facciamo un sacco di cose e trasportiamo sul dischetto posto in b: un vero laboratorio di attrezzi.

Uno di questi attrezzi, chiamato **autoexec.bat** e' anch'esso un supercomando che va autocostruito sul dischetto del sistema DOS. Ecco la procedura di introduzione fatta ancora con il comando **copy con:**, nella quale appaiono i comandi di cui e' costituita la catena:

```
A>    copy con: autoexec.bat
      date
      time
      graphics
      chkdsk
      pause
      gwbasic                                       [F6+return]
```

L'utilita' di avere generato e copiato sul **dischetto di lavoro** questo autoexec.bat risiede nel fatto che in questo modo, all'accensione della macchina con tale dischetto tutto e' automatico. La macchina chiede se vogliamo riassegnare la data e l'ora, si caricano i componenti **graphics** e **gwbasic** e si controlla lo spazio ancora disponibile (chkdsk). Nella figura B.2 forniamo un'illustrazione del meccanismo del supercomando mater.bat nella cui catena compare, insieme ai componenti DOS, anche l'altro supercomando autoexec.bat. Abbiamo quindi fabbricato un secondo livello di supercomandi...e il meccanismo potrebbe, ovviamente, non aver fine!

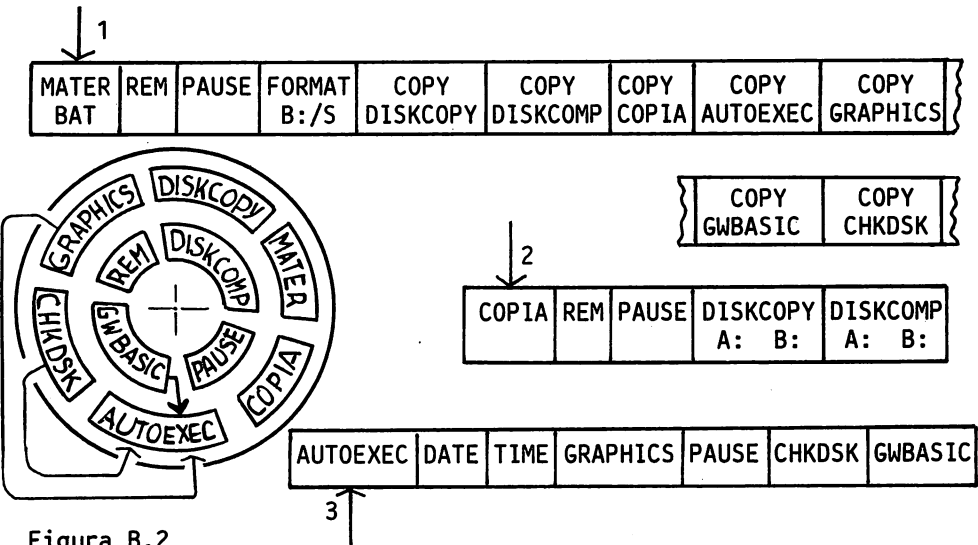



Figura B.2

In esecuzione, dichiarando **mater** si esegue la catena **1** che crea nel dischetto **μ** posto in **b:** una struttura di componenti DOS come quella riportata nella figura B.3.

```
A>dir b:

 Volume in drive B has no label
 Directory of  B:\

COMMAND  COM    15957   5-31-84   9:00a
DISKCOPY COM     2051   5-31-84   9:00a
DISKCOMP COM     2051   5-31-84   9:00a
COPIA    BAT      185   5-21-85   6:54a
AUTOEXEC BAT       46   5-21-85   6:58a
GRAPHICS COM      971   5-31-84   9:00a
GWBASIC  EXE    70160   6-29-84   9:00a
CHKDSK   COM     6468   5-31-84   9:00a
        8 File(s)   233472 bytes free
```

Figura B.3

In tal modo, quando sara' il dischetto **μ** a erogare le risorse DOS, si avranno le seguenti situazioni di utilita':

- il dischetto μ, grazie alla catena 3 (autoexec), sara' in grado di portare la macchina automaticamente fino al livello Basic chiedendo solo alcuni consensi (in pratica si puo' battere solo **return** fino alla comparsa dell'ambiente gwbasic [**Ok**];
- quando si vuole, battendo la parola **system** e tornando quindi in ambiente DOS [>], chiamando il supercomando **copia** e mettendo un dischetto vergine in **b:** si effettuera' la catena **2** che realizza appunto una copia in **b:** (con relativa verifica) del contennuto del dischetto posto in **a:**.

UN ALTRO ESEMPIO DI CREAZIONE DI SUPERCOMANDI

Obiettivo: Ottenere un dischetto **Σ** che contenga:

- command.com (il nucleo del DOS)
- autoexec.bat (echo off..mode lpt1:132,8 graphics gwbasic)
- mode.com
- graphics.com
- gwbasic.exe

Passo 1: Inserire un disco (che chiameremo Σ) in **b** e formattarlo, avendo il DOS in **a** [a> format b:/s]

Passo 2: Inserire il disco Σ appena formattato nel drive **a** e generare il seguente supercomando, tipo autoeseguibile:

```
A> copy con: autoexec.bat
echo off          (annullera' ogni risposta di sistema)
mode lpt1:132,8   (fara' stampare 16,6 car.per pollice)
graphics
gwbasic           [F6+return]
```

Passo 3: Rimettere il dischetto DOS in **a** e il disco Σ in **b** e copiarvi i seguenti tre componenti DOS:

```
A>  copy mode com b:
    copy graphics.com b:
    copy gwbasic.exe b:
```

Al termine verifichiamo con una **dir** il contenuto di Σ e vi troveremo:

```
COMMAND  COM   15957   5-31-84   9:00a
AUTOEXEC BAT      46   7-13-85   8:35p
MODE     COM    2382   6-29-84   9:00a
GRAPHICS COM     971   5-31-84   9:00a
GWBASIC  EXE   70160   6-29-84   9:00a
        5 File(s)   244736 bytes free
```

B.6.1 I SUPERCOMANDI PARAMETRICI

In un supercomando i singoli comandi della catena, anziche' riferirsi ai nomi effettivi di file, possono essere associati a simboli numerati (o parametri): nella fattispecie il simbolo usato e' [%] seguito da un numero compreso tra 1 e 9.

L'utilita' dei parametri nella catena del supercomando e' che possono evitare di ripetere i veri nomi a livello dei singoli comandi della catena: i veri nomi, poi, vanno dichiarati al momento dell'esecuzione subito dopo il nome del supercomando.

Ad esempio, supponiamo di dover generare un supercomando costituito di comandi che operano su due file, che vengono indicati simbolicamente come parametri %1 e %2, anziche' esplicitarne subito il nome.

Scriviamo poi un supercomando di nome **mix.bat** che esegua appunto le operazioni previste:

```
a> copy con:mix.bat

    type %1
    type %2
    copy %1+%2
    type %1
    type %2
```

Supponiamo di aver scritto due file di nome P e R contenenti rispettivamente Parigi e Roma. All'esecuzione di **mix** dobbiamo specificare, dopo il nome del supercomando, gli effettivi nomi dei due file che il supercomando assegnera' ai parametri secondo l'ordine di comparizione.

Ad esempio, se lanciamo il comando **mix r p** avremo un tipo di risultato che rispetta l'ordine in cui sono stati generati i comandi nella catena e quello in cui sono stati dichiarati i nomi dei file nell'argomento di **mix.** Siccome **r** viene prima di **p,** allora al parametro %1 corrispondera' **r** e a %2 **p.** Si compiranno percio' le seguenti operazioni:

- type r......ROMA

- type p......PARIGI

- copy r+p (secondo la logica della concatenazione dei file -vedasi il paragrafo successivo- l'operazione accoda il contenuto di p al contenuto di r)

- type r ROMA
PARIGI

- type p PARIGI

Il supercomando avrebbe funzionato nello stesso modo se anziche' dichiarare mix r p avessimo dichiarato **mix g t,** dove g e t fossero stati i nomi di altri file.

B.7 IL CONCATENAMENTO DEI FILE

Invece di costruire supercomandi che attraverso la dichiarazione di un unico nome eseguono una catena di comandi si puo' impiegare un comando singolo per concatenare dei file. Ad esempio un comando **copy** puo' farlo usando la seguente sintassi:

```
A>  copy r+s+t  b:qq
```

dove r, s, t sono i nomi dei file da concatenare e copiare, uno di seguito all'altro, in un unico file di nome **qq** localizzato nel dischetto posto in **b**. Vediamo come verificarlo. Creiamo tre file di nome r, s, t:

```
1)    A>  copy con r      CR
      nel mezzo           CR
      F6                  CR

2)    A>  copy con s      CR
      del cammin          CR
      F6                  CR

3)    A>  copy con t      CR
      di nostra vita      CR
      F6                  CR

4)    A>  copy r+s+t  q   CR

5)    type q              CR

      nel mezzo
      del cammin
      di nostra vita

6)    A>  type r          CR

      nel mezzo
      del cammin
      di nostra vita
```

Si noti che dopo la creazione dei tre file ai passi 1), 2), 3), abbiamo scritto il comando, al passo 4), che nella sintassi della concatenazione dei file, significa: accoda i file **s** e **t** al file **r** e il file risultante copialo col nome **q** sul dischetto posto nel drive attivo (a:). Le verifiche, fatte ai passi 5 e 6, confermano le operazioni avvenute: infatti, il file **q** e il file **r** hanno lo stesso contenuto. Un altro esempio interessante -ma il lettore ne puo' trovare di piu' pertinenti alle sue necessita'- e' il seguente:

```
A>  copy q + *.pag
```

che significa: tutti i file con estensione **.pag** vanno concatenati uno di seguito all'altro e accodati al file di nome **q**.

Creiamo, ad esempio tre file **f, g, h,** tutti aventi come estensione del nome il sottonome **.pag**:

```
A>  copy con f.pag   CR
mi ritrovai          CR
F6                   CR

A>  copy con g.pag   CR
per una              CR
F6                   CR

A>  copy con h.pag   CR
selva oscura         CR
F6                   CR

A>  copy q + *.pag   CR
```

A verifica del concatenamento avvenuto, basta scrivere **type q** **CR** e otterremo:
"nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura".

INDICE ANALITICO

Volete mettere in pratica, su M24, tutti i programmi contenuti in questo libro? Volete modificare ed esaminare a vostro piacimento oltre 170 programmi?

Potrete avere il dischetto (con istruzioni) relativo a tutti i 170 programmi contenuti nel libro *"Un personal computer MS-DOS. Impariamo a programmare con M24"* ordinandolo contrassegno di lire 30 000 a:

TELEIA Società generale d'informatica
Via Telesio, 22
20145 Milano

Per ulteriori informazioni telefonare al numero:
02-43 29 64-46 93 397

* Il prezzo del dischetto potrà subire variazioni a partire dal 1 gennaio 1987.

Vi prego di inviarmi contrassegno il dischetto relativo a:
"Un personal computer MS-DOS. Impariamo a programmare con M24".

____
nome e cognome

____
indirizzo

____
codice postale e città

Da spedire in busta chiusa.

CSC/Collana di Scienza dei Calcolatori

Giampiero Bianchi
UN PERSONAL COMPUTER MS-DOS
Impariamo a programmare con M24

Il personal computer è sempre più diffuso in diversi settori,
dal mondo del lavoro a quello della scuola;
è quindi sempre più pressante la domanda di poter usare,
subito e al massimo delle sue possibilità,
la macchina che abbiamo di fronte.
A questo tipo di domanda risponde questo manuale
in cui il lettore è guidato, in modo chiaro e ricco di esempi,
alla costruzione di tutte le attività che la macchina può svolgere.
Una descrizione delle caratteristiche generali
e dell'hardware di M24, dettagliate informazioni sui comandi del DOS,
tecniche sempre più raffinate per trattare dischi e file,
un approfondito esame del mondo della grafica
e delle sue enormi potenzialità,
la musica, molti programmi di utilità: tutto questo permette
al lettore, partendo da problemi semplici e ben formalizzati,
di arrivare alla soluzione dei molti problemi concreti
che quotidianamente gli si pongono.

BIANCHI*PERSONAL C MS-DOS M24

Prezzo al pubblico L. 22 000
I.V.A. inclusa